ABSTRACT

Title of Dissertation: NETWORK AND DOMAIN AUTOCONFIGURATION: A UNIFIED FRAMEWORK FOR LARGE MOBILE AD HOC NETWORKS

Kyriakos Manousakis, Doctor of Philosophy, 2005

Directed By: Professor John S. Baras
Department of Electrical and Computer Engineering

Configuration management is critical to correct and efficient operation of large networks. In those cases where the users and networks are dynamic and ad hoc, manual configuration quickly becomes too complex. The combination of the sheer number of nodes with the heterogeneity and dynamics makes it almost impossible for the system administrator to ensure good configuration or even ensure correct operation. To achieve the vision of pervasive computing, nodes must automatically discover their environment and self-configure, then must automatically reconfigure to adapt to changes.

Protocols such as DHCP, DDNS and mDNS provide some degree of host autoconfiguration, but network administrators must still configure information such as address pools, routing protocols, or OSPF routing areas. Only limited progress has been made to automate the configuration of routers, servers and network topology. This dissertation proposes the autoconfiguration of most host, router and server information, including the automatic generation and maintenance of hierarchy, under the same architectural, algorithmic and protocol framework. The proposed unified framework consists of modules (DRCP, DCDP, YAP, ACA) responsible for the

entity autoconfiguration and from a modified and well adjusted general optimization (Simulated Annealing) based algorithm for the domain autoconfiguration. Due to the generality of the optimization algorithm, the generated hierarchy can improve dynamically selected network performance aspects represented by appropriately designed objective functions and constraints. An indicative set related to the physical characteristics of the domains and node mobility is provided.

Even though SA has been adjusted for faster convergence, it may still be unable to capture the dynamics of rapidly changing networks. Thus, a faster but suboptimal distributed hierarchy generation mechanism that follows the design philosophy of SA-based mechanism has also been introduced.

Inevitably, due to network dynamics, the quality of the hierarchy will degrade. In such scenarios, the frequent reapplication of the expensive optimization based hierarchy generation is prohibitive. Hence, for extending the domain formation framework, distributed maintenance mechanisms have been proposed for reconstructing the feasibility and quality of the hierarchy by enforcing localized decisions.

The proposed framework has been applied to provide solutions on some realistic network problems related to hierarchical routing and topology control.

NETWORK AND DOMAIN AUTOCONFIGURATION:
A UNIFIED FRAMEWORK FOR LARGE MOBILE AD HOC NETWORKS

by

Kyriakos Manousakis

Dissertation submitted to the Faculty of the Graduate School of the
University of Maryland, College Park, in partial fulfillment
of the requirements for the degree of
Doctor of Philosophy
2005

Advisory Committee:
- Professor John S. Baras, Chair
- Professor Richard J. La
- Professor Armand M. Makowski
- Dr. Anthony J. McAuley
- Professor A. Udaya Shankar

To my wonderful parents, Athanasia and Konstantinos

my beloved sisters, Ariadni and Artemis

and to my inspiration, Maria

Acknowledgements

First and foremost I would like to thank my advisor Dr. Jonh S. Baras for giving me the opportunity to collaborate with him. Without his guidance, support and vision this dissertation would have not been made possible. Apart from his extraordinary technical skills, and his deep knowledge of the professional world, he provided me with invaluable experience and views related to various aspects of professional life that have contributed significantly to the development of my career. I will always be grateful to him for his generosity, for having faith in my abilities and for helping me make my first significant research experience extremely enjoyable.

I owe a debt of gratitude to Dr. Anthony J. McAuley, my mentor at Telcordia Technologies and friend, with whom I have collaborated all the way throughout this challenging journey. Anthony's vast knowledge on networking and his unstinting support and encouragement throughout the years helped me grow both mentally and professionally. I shall always treasure his commitment to research and his willingness to patiently guide me through this marathon endeavor.

I am thankful to my various Telcordia Technologies colleagues and supervisors, who I was fortunate enough to meet and collaborate with during my several internships there, gaining not only technical skills but also many friends. Special mention must be made to Dr. Raquel Morera and Dr. Ken Young, who have always exhibited tremendous understanding and provided me an extraordinarily supportive work environment. I can truly say that, but (if it was not) for their support and cooperation, I would not have successfully completed this dissertation.

I would like to thank Dr. Richard J. La, Dr. Armand M. Makowski, Dr. Anthony J. McAuley¸, and Dr. A. Udaya Shankar for reading and commenting on my dissertation, and also for the honor they gave me by accepting to take part in my committee.

My special thanks, for all the memories and laughs we had, to my roommates and friends in Stewart Court – Theodoros Salonidis, Iordanis Koutsopoulos, Nikos Kanlis and Kostas Tsoukatos, who helped me settle in this place and made me feel like home. I cannot forget mentioning my officemates and friends, Theodoros Salonidis, Majid Raissi-Dehkordi and Svetlana Radosavac, who kept me company the long days and nights in AVW3187. During my stay in Maryland, I feel blessed having made many good friends and having wonderful memories, especially from the Graduate Hellenic Association DIGENIS and the soccer group from the ECON department of University of Maryland, College Park. They are so many that it would not have been possible to mention each one of them individually but they will always have a special place in my memories.

Towards the way to this achievement, I was fortunate enough to meet two people that have always been a huge source of inspiration to me Mr. Michalis Karavolias and Dr. Michalis Paterakis. The first one showed me the magic and harmony of mathematics and the second one the beauty and precision of engineering.

I would like to thank Maria Striki for her patience, support and most importantly love. She was there for all the good and bad times encouraging me to continue pursuing any difficult task coming in my way, even this of losing 150 lbs. I will be always grateful to her.

Words cannot express my most sincere gratitude to my wonderful parents Athanasia and Konstantinos and my sisters Ariadni and Artemis. Without their love, endless support, and understanding, this would not be possible. They are the main reasons I have been able to reach this point. This dissertation is dedicated to them.

This work was supported through the collaborative participation in the Communications & Networks Consortium sponsored by the U. S. Army Research Laboratory under the Collaborative Technology Alliance Program, Cooperative Agreement DAAD-10-01-2-0011.

Table of Contents

List of Tables

List of Figures

Chapter 1: Introduction

1.1 Introduction

Configuration management is critical to correct and efficient operation of large networks. In those cases where the users and networks are dynamic and ad hoc, manual configuration quickly becomes too complex. The combination of the sheer number of nodes with the heterogeneity and dynamics makes it almost impossible for the system administrator to ensure good configuration or even ensure correct operation. To achieve the vision of pervasive computing, nodes must automatically discover their environment and self-configure, then must automatically reconfigure to adapt to changes. Protocols such as DHCP, DDNS and mDNS provide some degree of host autoconfiguration, but network administrators must still configure information such as address pools, routing protocols, or OSPF routing areas. Only limited progress has been made to automate the configuration of routers, servers and network topology. We propose the first unified attempt to combine both the self configuration of much of the host, router and server information, together with the automatic generation and maintenance of hierarchy under the same algorithmic framework. Testbed implementations show the approach is practical, while analysis reveals its scalability, rapidness and efficiency with respect to network performance.

Future commercial, military and emergency networks require changes to traditional network management. Configuration management, in particular, must ensure correct and efficient network operation through setting parameters such as:

- IP Addresses of an interface.
- Network Parameters (e.g., Default MTU size).
- Server addresses (e.g., for DNS or Certificate Authority server).
- Routing information (e.g., default route or routing protocols).
- IP Address pools (e.g., for DHCP or MADCAP server).
- Security keys.

While protocols such as DHCP, DDNS and mDNS have allowed more autoconfiguration, network administrators must still manually configure much of this information. We need new protocols that are able to configure all these parameters especially in routers and servers.

In many cases configuration management must also construct hierarchies (e.g., routing areas and security domains) for scalability, efficiency and manageability. Today, the construction of hierarchies is a manual process, performed off-line by experts, because it requires difficult optimizations. For example, in the creation of OSPF areas in dynamic networks, the savings in reduced routing overhead must be balanced with the overhead of hierarchy maintenance and mobile node reconfiguration. Figure 1.1.1 shows an example of how OSPF areas, with aggregation at area boundaries, reduce the number of OSPF LSA packets in a network (routing overhead). When all nodes are placed into one area, the routing overhead grows quadratically with the number of nodes (n); however, with a two-level hierarchy with $\sqrt{n}$ nodes per OSPF area, overhead grows much less rapidly. This does not mean, however, that, for example, a 25 node network should be divided into 5 domains with 5 nodes in each domain. The configuration management must take into account the increased hierarchy maintenance and mobility management overheads. Further adding to complexity, Figure 1.1.1 shows that the lowest routing overhead is achieved by keeping nodes with similar velocities into the same areas.

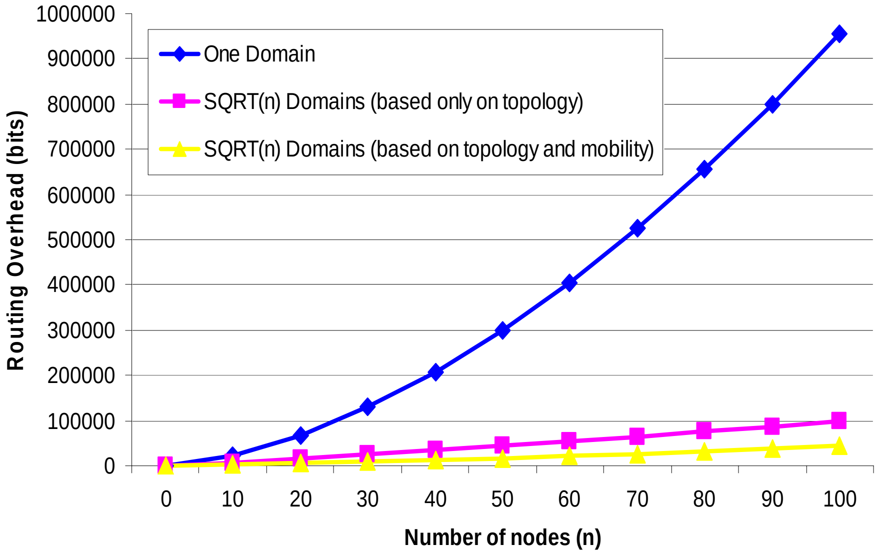


Figure 1.1 Reducing routing overhead using a two level hierarchy

We believe configuration management must be **cheaper** (e.g., more plug and play), **more robust** (e.g., no human intervention), **faster** (e.g., in seconds), and **better optimized** (e.g., to the link error rate) than current approaches. Moreover, configuration management must be able to deal with more **network dynamics** (e.g., varying topology) and be more **scalable** (e.g., to support 10K nodes). In some cases the configuration must be done with little or no fixed infrastructure.

This dissertation proposes the autoconfiguration of most host, router and server information, including the automatic generation and maintenance of hierarchy, under the same architectural, algorithmic and protocol framework. The framework that will be presented and analyzed throughout the dissertation, consists of two parts, the communication part and the decision making part. The decision making part is responsible for obtaining the appropriate network and hierarchy configuration with respect to the imposed network performance objectives and the communication part is responsible for the distribution of the configuration decisions and the collection of the required information utilized from the decision making part of the framework. Namely, the modules that constitute the communication part of the proposed framework are the modules:

- Dynamic Configuration Distribution Protocol (DCDP)
- Dynamic and Rapid Configuration Protocol (DRCP)
- Yelp Announcement Protocol (YAP)

These modules are part of the introduced IP Autoconfiguration Suite (IPAS), which is responsible for the network configuration.

Furthermore, the decision making part is based on an enriched, modified and well adjusted SA optimization (SA is a randomized general approximation algorithm, which asymptotically behaves as a global optimization algorithm – provably SA asymptotically obtains the global optimal) algorithm. The selection of a general optimization algorithm for obtaining configuration decisions is justified from the philosophy behind the proposed general configuration framework. The generality of the framework is with respect to the performance objectives imposed to the network. These objectives can be selected dynamically or change during the lifespan of the network. The same framework can obtain dynamically the appropriate network and hierarchy configuration that satisfies the new performance objectives, without the need to utilize a different set of algorithmic modules and functions, which are tailored to this new set of objectives.

The biggest challenge on the network and hierarchy framework design is the design of lightweight, scalable, robust and efficient communication modules and the adaptation of the optimization (SA) algorithm to the dynamic nature of the network environments (i.e. mobile wireless ad hoc networks) under consideration, by reducing its convergence time considerably without affecting its high quality optimization ability. On the same lines, another challenge is the ability of SA to handle efficiently the dynamics of the network, since its continuous reapplication for every topological change is not suggested due to the overhead and latency costs involved. Thus, the decision making mechanism has been enriched with suboptimal distributed (localized) heuristics, which have been designed following the same spirit of generality and independence of the performance objectives imposed.

The detailed description and analysis of the proposed unified network and hierarchy configuration framework are provided in this dissertation. Specifically, chapter 2 presents the IP Autoconfiguration Suite (IPAS) and its modules, which are responsible for the network configuration and additionally provide communication capabilities utilized from the general configuration framework. Chapters 3 and 4 discuss the introduced dynamic hierarchy generation mechanism. This discussion involves the presentation of the cornerstone algorithm for the decision making part of the configuration framework. Namely, the functionality and properties of the general randomized approximation algorithm (SA) are provided along with its most significant parameters responsible for its performance. Furthermore, the indicative set of the metrics, cost functions and optimization constraints utilized for the description and evaluation of the hierarchy generation framework are being provided. Chapter 5 deals with the detailed description of the techniques enforced for the enrichment, modification and adaptation of the SA algorithm on the dynamic nature of the wireless mobile ad hoc networks. The main objective of the latter techniques is the improvement of convergence time of the SA algorithm without penalizing its optimality. Even though the performance characteristics of the cornerstone algorithm (SA) were adjusted appropriately, in cases of rapidly changing networks, the algorithm might not be able to capture their dynamics. Thus, a suboptimal distributed generation mechanism that follows the design philosophy of the SA-based mechanism is being introduced and evaluated in chapter 6. The hierarchy generation framework attempts to optimize the imposed hierarchical structure by involving the entire network on the decision making process. Due to the dynamics of the network, this is not practical (expensive and slow) in cases of frequent and localized changes. Hence, chapter 7 presents and analyzes the localized (distributed) maintenance mechanisms introduced for providing to the network the ability of adapting to the changes by acting and rebuilding the optimality of the hierarchical structure locally. Finally, chapter 8 applies the proposed framework on some realistic network problems related to hierarchical routing and topology control for optimizing the power consumption by adjusting appropriately the transmission power.

Chapter 2: Autoconfiguration of MANETs

2.1 Introduction

The networking capabilities and the number of devices that will constitute the future commercial, military and emergency networks suggest change in the spirit of their traditional management. One aspect of this management is the configuration of these devices so that they can be part of the network. The most important requirements of the new way of configuring networks are the speed of configuration (e.g., large number of nodes can be configured and communicate in few seconds) and the dynamic nature of the management (e.g., human intervention is not anymore required). Traditionally, the network manager was responsible for the configuration of the various network entities and this is a costly procedure with respect to the required time and human effort. The future network systems indicate that this amount of time or human resources may not be available so the network and the various entities must have the intelligence to configure themselves.

The problem becomes even more complicated and significant when the network environments under consideration are dynamic (e.g., varying topology) due to area conditions and node mobility. Even this is the most difficult interpretation of the autoconfiguration problem; this is the most interesting one. The future networks, especially the military and the emergency networks have been mainly envisioned to consist of large number of wireless mobile devices, in topological areas with obstacles and high interference characteristics.

2.2 Related Work

The problem of network autoconfiguration is very important for the general acceptance and correct functionality of MANETs. Due to the characteristics of this type of networks, the static configuration (i.e. pre-assigned IP addresses) of the participating nodes is meaningless. The topology of the network is changing dynamically and part(s) of it may become unavailable from time to time. The importance of this problem has been understood from the researchers and studies have been performed for the design of efficient autoconfiguration solutions. In this section we present the various approaches that have been proposed.

Network autoconfiguration is a new problem for network environments like the MANETs but is not new for the INTERNET community. There, the efforts for an efficient solution have so far focused on the more limited objective of autoconfiguring static hosts and small networks. Subnet configuration protocols, such as PPP and DHCP [1][2][3][4] allow clients (hosts) to dynamically request an address and other configuration parameters when they first establish a communication link. Due to the static nature of the networks under consideration, the solutions (DHCP, PPP) are based on a client- server protocols. For example, in order a host to get configured requires an online DHCP server, which is pre-assigned from the network administrator to provide configuration information to the hosts requesting it. In networks like MANETs, a dedicated configuration server may not be available all the time. Also, DHCP can configure only hosts but is not useful for router nodes. By definition all nodes in a MANET are considered routers, so DHCP cannot handle this type of nodes. So, solutions from the hardwired network world cannot be applied in dynamic and infrastructureless environments.

On going research for dynamic networks' autoconfiguration is being performed by a dedicated IETF Working Group called Zeroconf, which mainly focuses on environments that lack online configuration servers. The solutions provided by Zeroconf are not directly applicable to MANETs due to the type of dynamic networks they consider. The study performed by Zeroconf working group focuses on single segment networks, where all the participating nodes can communicate directly through link-layer broadcasts/ multicasts or multiple such networks which are connected on the same router. Obviously, these types of networks are subcategories of MANETs. The latter networks as opposed to the Zeroconf networks are considered multihop networks, so link level broadcasts do not guarantee their reception from all the participating nodes. Moreover, the Zeroconf approaches for Duplicate Address Detection (DAD) cannot be applied in MANETs.

The problem is far more complicated and demanding compared to hardwired and Zeroconf networks and has been categorized [5] in the following four subproblems:

- Address Autoconfiguration

  Address configuration involves the configuration of network interfaces with unique addresses and the selection of the appropriate subnet mask to be used. The subnet mask identifies the network address and, among other things, allows an IP stack to determine whether it can deliver a datagram directly. Furthermore, and due to the dynamics of the networks under consideration, address configuration mechanisms should be able to detect duplicate address assignment and cope with the collisions of this kind.

- Name-to-Address Translation

  IP applications typically identify endpoints by name rather than by address. This provides operational stability when the address of the endpoint changes, since the name will remain the same. From the name-to-address translation mechanisms, it is required that the IP address to be obtained is associated with a name and the selection of the name is associated with an IP address.

- Service Discovery

  Clients should be able to discover services on the network without prior configuration, and without any administered configuration management services (such as directories) on the network. Furthermore, the service discovery mechanism has to be lightweight in terms of the overhead imposed into the network (e.g. must no cause broadcast storms or other non scalable behavior). There are two categories of services, the indistinct and the distinct ones. In the indistinct services any server will perform the exact same function, as opposed to the distinct services where the service provided depends on the server that will be contacted. Indistinct services include services like DNS, Web proxies, SMPT relays and examples of distinct services are the IP-enabled printers, file servers, non-replicated databases. The network entities have to be able to find and contact the server that best meets their needs.

- Multicast Address Allocation

Some multicast applications require a unique multicast address to prevent other applications from conflicting with them. A multicast address conflict can cause the applications to fail. The assignment of multicast addresses to applications is analogous to the assignment of unique addresses to the network entities, where address conflict must be prevented. The Zeroconf working group has introduced the Zeroconf Multicast Address Allocation Protocol (ZMAAP), which allocates unique addresses to multicast applications, prevents the reallocation of already assigned addresses and notifies the applications in case of multicast address collision.

The subproblem of address configuration is the most important, since it is the essential step for a network interface to become part of the communication network. Most of the studies being performed are related to this problem. The existing solutions can be classified [6] into three large categories. These categories are:

- Conflict Detection Allocation
- Best Effort Allocation
- Conflict Free Allocation

Many of the existing address assignment protocols belong into the first category (conflict detection allocation). The main characteristic of these protocols is that the non configured network entities are assigned an address and then the duplicate address detection module checks if there is a collision with another node, by requesting the approval of this assignment from the configured nodes of the network. If the conflict is found by veto from a node with the same address, the procedure is repeated until there is no collision detected. The main differences between the protocols of this category are the selection of the address to be assigned and the mechanism they apply for duplicate address detection (DAD). In the conflict detection allocation category belongs the protocol proposed in [7], which is based on the adaptation of the stateless IETF Zeroconf autoconfiguration protocol for MANETs [8]. A node randomly chooses an address and performs a DAD by flooding the network with an address request (AREQ) message, which contains the selected address. A node having the same address defends it by replying with an address reply (AREP) message, which is sent over the reverse path established by the AREQ message. If there is not another node with the same address in the network, a dedicated timer expires at the originator and the address is considered unique. The DAD mechanism proposed by [7] is query-based. The drawbacks of the approach are that network merging is not supported and the DAD mechanism is not scalable due to the overhead imposed by the flooding.

Another approach presented in [9], which also belongs to the conflict detection allocation category, is based on the Weak DAD (WDAD) mechanism. WDAD is integrated with the routing protocol and can continuously detect duplicate addresses due to the information added and carried from the underlying routing protocol. This mechanism requires modification of the routing protocol packets format, where a key related to each address is added. The key can be of arbitrary length and is chosen once by each node either randomly or with respect to a Universal Unique ID (UUID). A node detects a conflict if it receives two address-key pairs with the same address, but different keys. Obviously, a collision cannot be detected if two different network entities choose the same address and the same key. In the case of randomly selected keys, the probability of something like that happening decreases with increasing key length. On the other hand, increasing the key length imposes extra overhead on the routing protocol. So, there is a trade-off between non-detectable collisions and overhead.

In the proposed algorithms of the best effort allocation category, the nodes responsible for assigning addresses to the non configured ones, try to allocate unused addresses based on their knowledge of the set of addresses being assigned so far. At the same time the new node utilizes conflict detection to guarantee that the assigned address is free. The protocols presented in [10] and [11] are the best representatives of the best effort allocation category. In the protocol proposed in [10] each configured node is able to assign addresses to new nodes and therefore maintains an allocation table of already assigned addresses in the network. A new node called “requester” searches for an already configured node called “initiator” be sending a special broadcast message. The “initiator” chooses an unassigned address with respect to its addresses allocation table, and ensures the uniqueness of this address by a mutual exclusion algorithm, where it requests from all the configured nodes to approve this selection and mark this address as allocated on their address allocation tables. If all the configured nodes reply positive about the address then the “initiator” commits and assigns the address to the “requester”. In the case where there are non-replying nodes, the “initiator” after a number of retries assumes that these nodes have left the network and removes their addresses from its allocation table. This protocol is claimed that can handle successfully network merges by identifying each partition with a unique ID. This ID is composed of the smallest address in the network and a Universal

Unique ID (UUID) that is provided by the node with the smallest address. The partition ID is advertised periodically, so when nodes receive such messages with different partition ID they assume that the two partitions have merged. In this case the nodes exchange their allocation tables. The nodes that find that their addresses in the allocation table of the other partition, have to give up their addresses. The drawback of the protocol is that bases its functionality on global states (address allocation tables). In order the method to be successful, these tables have to be maintained updated, which requires reliable message exchange (reliable broadcast).

The other protocol (PACMAN) that belongs in the best effort allocation category is presented in [11] where the addresses are assigned in probabilistic way to the non configured entities. A passive DAD (PDAD) mechanism, which relies on address allocation tables and the routing protocol, is utilized to check for collisions. A node running PACMAN assigns an address to itself using a probabilistic algorithm. Based on a pre-defined collision probability, an estimation of the number of nodes and an address allocation table, the algorithm calculates the size of the virtual address space, randomly selects an address from this space. If with respect to the local address allocation table, the address has not been assigned before, the node immediately gets configured with this address. The calculation of collision probability is computed in analogy to the well known birthday paradox. Since the address allocation table may be out-of-date, a passive DAD (PDAD) mechanism investigates the existence of a collision. The DAD mechanism is passive because it relies on the processing of the routing protocol messages received by the node. In [11] many types of PDAD mechanisms are proposed, depending on the underlying routing protocol. These mechanisms demand global clock synchronization and their functionality is based on various assumptions (i.e. routing information received from each node can never be older than a time span t_d, which can be estimated accurately). PACMAN is heavily depends on the underlying routing protocol, which has to be present to the non configured network and belong to the proactive class of routing protocols. In case of reactive routing protocols the approach cannot be applied, since there is not periodic exchange of information that can be utilized for the update of address allocation tables and the functioning of PDAD mechanism.

Last but not least is the category of conflict free allocation protocols. These protocols assign unallocated addresses to the new nodes. The latter is achieved by the assumption that the nodes taking part in the configuration process have disjoint address pools. Thus they can be sure that the addresses to be allocated are different. The more representative protocols of this category are presented in [12] and [6]. In [12] the disjoint pools of unallocated addresses are maintained based on the idea of binary splitting, presented initially in [13]. They extend binary splitting by relying on the binary buddy system for managing the pools. Buddy systems [14] are a type of segregated lists that support an efficient kind of splitting and coalescing. Binary buddies are the simplest and best known kind of buddy system. In this scheme, all buddy sizes are a power of two, and each size is divided into two equal parts. As in [13], these parts represent the unallocated addresses and are distributed throughout the network, so that can be utilized for assigning addresses to the non configured network entities. The assigned addresses are removed from these sets, so that they contain only non-assigned addresses. Due to this property, DAD is obsolete for the conflict free allocation protocols. The main assumption of [12] is that there are no conflicts on the initial pool(s) of addresses. A weakness is that in scenarios where two previously unrelated networks merge, the assumption might not hold, and address conflicts may exist.

The idea presented in [6] belongs also in the category of conflict free allocation protocols. The main idea differs from the previous one, since the conflict free allocation is not based on splitting the pools into disjoint parts but on a function $f(n)$ that generates sequences of different integers in a range *R*. The sequences of $f(n)$ satisfy the following two properties, if *R* is large enough:

- The interval between two occurrences of the same number in a sequence is extremely long.
- The probability of more than one occurrence of the same number in a limited number of different sequences initiated by different seeds during some interval is extremely low.

Initially the first node A selects a random number as its address and a random value as a seed for $f(n)$ (e.g. the value of this function at each instance at every node is called the "state" of the corresponding node). When another node B requests to be configured from A, then A utilizes $f(n)$ to generate an address for B and a seed to be used from B for its $f(n)$ and also A updates its state. In that fashion the network gets configured with addresses from the range *R*. Node A is called the "prophet" since it knows in advance which addresses are going to be allocated and which of them will collide. In the latter case it can initialize local conflict detection before the allocation of the corresponding addresses. The scalability and the correctness of [6] depend on the effectiveness of $f(n)$, which has to produce very long conflict free sequences of integers. Problems can also appear in the case where multiple isolated nodes appear into the network simultaneously and get initialized with the same random numbers. In such cases the algorithm fails, since the participating nodes do not have the means to detect this phenomenon.

The following table has been presented in [6] and provides some interesting comparison highlights between the three categories of configuration (address assignment) algorithms. For the communication overhead and latency complexities of the various algorithms, it can be assumed that the number of mobile nodes is *n*, the number of links is *l*, the average transmission time is *t*, the network diameter is *d* (in terms of nodes) and the retry time is *k*.

	Conflict Detection Allocation	**Best Effort Allocation**	**Conflict Free Allocation**
Network Organization	Flat / Hierarchical	Flat / Hierarchical	Flat
State Maintenance	Stateless	Stateful	Partially Stateful [12] Stateful [6]
Address Conflict	Yes	Yes	No
Address Reclamation	Unneeded	Needed	Needed
Complexity	Low	High	Low
Communication Overhead	$O((n+l)\times k)$	$O((n+l)\times k)$	$O\left(\frac{2l}{n}\right)$
Latency	$O(2\times t\times d\times k)$	$O(2\times t\times d\times k)$	$O(2t)$
Scalability	Low	Low	High

Table 2.1. Comparison Highlights between the various address assignment approaches

The approach presented in this dissertation belongs in the conflict free allocation category, since it bases its functionality on splitting and distributing disjoint pools of available addresses throughout the network. When a non configured node requests an address then by selecting and assigning addresses from these pools, it is certain that there is no collision with a previously allocated address. Details on the approach are presented in later sections in this chapter.

2.3 Problem Description

Even though the autoconfiguration of MANETS can be described easily as the problem of providing the various network entities with the appropriate information so that they can become active members of the communication network, the internal specifications and requirements imposed by the problem are far more complicated. What is the appropriate information required by a network entity to become part of the communication network and how this information can reach this entity, are the most important questions we have to answer.

The problem becomes even more difficult and more realistic when we consider multihop networks where configuration information has to traverse paths that are larger than one hop to reach the non configured nodes. The routing of this information might be required to happen without the involvement of routing protocol and probably through nodes that do not possess an IP address. These are issues that will be addressed from the solutions provided in this dissertation. As we have mentioned the problem of autoconfiguration consists of many sub-problems due to the nature of the network, where many entities and modules have to be configured (i.e., IP addresses, DNS, routing, gateways) so that we can exploit its correct and full functionality. Among the various sub-problems the most important one is the conflict free assignment of IP addresses to the participating network entities. The importance of the assignment of IP address is that the functionality of every networking module in the TCP/IP stack is based on the individual address of the various network entities. If the assignment can happen efficiently and correctly across the network then this will decrease the level of difficulty for the configuration of the remaining networking modules and entities.

One of the existing approaches is based on the Dynamic and Rapid Configuration Protocol (DRCP). The limitation of DRCP as a standalone module is that can configure only network entities that belong on the same link. Since the solution we want to provide focuses on layer 3, the network environments under consideration consist of multiple links and for that reason we had to extend the DRCP module. We have suggested the Dynamic Configuration Distribution Protocol (DCDP), which extends the functionality of DRCP by interacting with the latter. The functionalities of the two protocols had left intentionally orthogonal, so that can be easily separated and applied to any future autoconfiguration module that requires their support. The next section gives a brief overview of DRCP, and in section 5 we describe DCDP, which is our contribution to the developed IP autoconfiguration suite.

The rest of this chapter consists of sections 6, where we provide a complete overview of the IP Autoconfiguration Suite (IPAS) and section 7, where we analyze and evaluate DCDP protocol. The presentation of network autoconfiguration solutions and their evaluation is being conducted in section 8.

2.4 Dynamic and Rapid Configuration Protocol (DRCP)

This section describes an IP link autoconfiguration protocol, called the Dynamic and Rapid Configuration Protocol (DRCP). Like the popular Dynamic Host Configuration Protocol (DHCP) on which it is based, DRCP configures nodes on a single link; however, DRCP also adds many features critical to future wireless dynamic networks [15]. Although now being developed for commercial 3G wireless networks [16], DRCP maintains the features needed for dynamic battlefield networks, such as allowing all nodes to be servers.

Designed for IPv4 and IPv6, DRCP is a link autoconfiguration protocol for wireless environments. DRCP is focused solely at layer 3 (L3). The assumption is that layer 2 (L2) protocols autoconfigure nodes into IP links, where each node can be reached in one IP hop from any other node in the same link. Also, another assumption on the functionality of DRCP is that the autoconfiguration protocols should be independent of the routing and mobility management protocols.

Although based on DHCPv6, DRCP has many extensions to:

- Provide rapid client configuration and reconfiguration after a move
- Make efficient use of wireless bandwidth
- Not require clients to broadcast to another clients

Also, a DRCP server can configure all interfaces on a link, including its own and those of any routers on the link. All DRCP nodes have a management interface for applications such as dynamic link reconfiguration and configuration management.

A node running DRCP is initially assumed only to know, which of its interfaces configure using DRCP. If there are multiple interfaces may be configured by DRCP, others using alternative techniques (e.g., using locally stored addresses or using DHCP).

All DRCP nodes run the same code: there is no separate client and server program (as there are in DHCP). After boot-up, a node assumes all its DRCP interfaces are configured to be DRCP clients. As a client the interface will attempt to discover a DRCP node acting as the DRCP server on the corresponding link. After a random time, however, interfaces that are not configured (e.g., cannot locate a node acting as a DRCP server) will check if they can become a DRCP server. A node becomes a DRCP server for an interface when it has configuration information, including a pool of available IP addresses. A DRCP node does not get this information through DRCP, but from:

1. Preset information (i.e., configuration file)
2. Management interface (i.e., from a configuration manager)

If a node becomes a DRCP server, then it will take the first available address from its address-pool and other configuration information to configure its own interface for that link. The node is then ready to serve other nodes on that link. A node can act as a DRCP server on a subset of its interfaces and as a DRCP client on another subset of them. Whether a node acts as a DRCP server or DRCP client for an interface can change due to the dynamic changes that will happen into the network overtime.

2.4.1 DRCP Client-Server Messages

DRCP-to-DRCP messages between clients and servers use UDP as its transport protocol. All messages from a client are sent to the well known DRCP_SERVER_PORT port. All messages from a server are sent to the

DRCP_CLIENT_PORT port. Although there are some differences between DRCP for IPv6 (such as the use of multicast or broadcast), we will describe DRCP generically.

DRCP messages have similar names and meanings to DHCPv6, but with important differences in their operation and syntax. The four principal messages are:

- **DRCP_SOLICIT:** Broadcast or multicast by a client to locate servers. There is no requirement for this message to reach all clients on the link.
- **DRCP_ADVERTISE:** Broadcast, multicast or unicast by servers to advertise their location, either periodically or in response to a DRCP_SOLICIT.
- **DRCP_REQUEST:** Unicast by clients to request configuration parameters from a server or to extend the lease on an address.
- **DRCP_REPLY:** Unicast by servers responding clients to a DRCP_REQUEST or DRCP_RELEASE message. When responding to a DRCP_REQUEST, the message contains the client's new configuration parameters.

DRCP also has the following messages:

- **DRCP_RELEASE:** Unicast by clients to relinquish an IP address and cancel remaining lease
- **DRCP_RECONFIGURE:** Unicast or multicast by server to offer client new configuration information (the client is assumed not to be in sleep mode).
- **DRCP_RESET:** Unicast or multicast by a server to reset the client (the client is assumed not to be in sleep mode).

2.4.2 Basic Call Flow

The basic operation of DRCP can be demonstrated by assuming the simple network of two nodes as it is shown in figure 2.1. The DRCP process is running on both Node *A* and Node *B* for the configuration of their interfaces on Link *x*. Initially both Node *A* and Node *B* listen as clients and send out DRCP_SOLICIT messages on Link *x*. Since both nodes have not been configured they cannot claim the role of the DRCP server on the link. Assume that Node *B* has a pool of available IP addresses and other configuration information; therefore, after checking that there are no other servers, it will become a DRCP server for Link *x*. Once it is a server, Node *B* will configure its own interface and start sending out periodic DRCP_ADVERTISE messages.

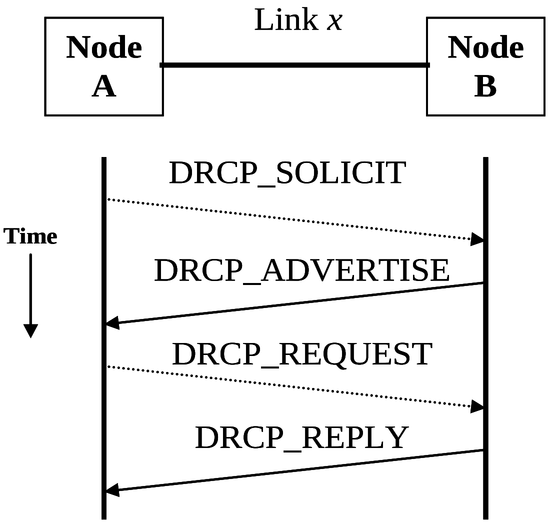


Figure 2.1. DRCP basic call flow

In figure 2.1 the time axis starts when Node *A* first becomes active (or moves onto Link *x*). If the latter node has not seen a DRCP_ADVERTISE message (either gratuitously or in response to the DRCP_SOLICIT message). Node *A* sends a DRCP_REQUEST to Node *B* requesting configuration information. Node *B* sends a DRCP_REPLY with configuration information, which Node *A* can immediately use to configure its interface on Link *x*.

Even though the participating nodes utilize the same DRCP module, they can behave as servers or clients onto a particular link, depending on the pre-configuration information they had acquired and the status of the configuration onto the corresponding link. The basic functionality of clients and servers is described briefly in the following two sections.

2.4.3 Basic DRCP Client Operation

The description of the DRCP client operation is best described by referring on Figure 2.2.

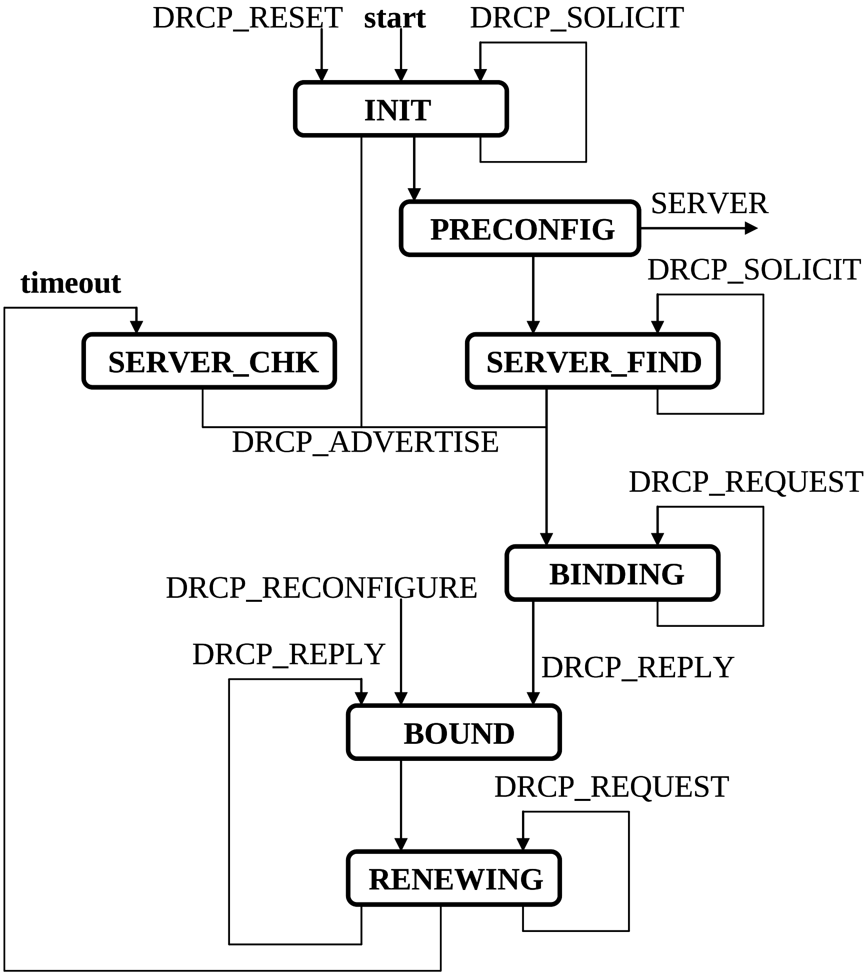


Figure 2.2. Simplified DRCP Client State Diagram

Initially all non-configured nodes behave as DRCP clients. A client starts in the INIT state and waits to hear DRCP_ADVERTISE messages on the DRCP_CLIENT_PORT. If it hears no messages, then the client may broadcast (or multicast) a DRCP_SOLICIT message to discover DRCP server nodes. Although the message is broadcast, it is not required the message to be received from all DRCP clients on the link. The latter assumption is important for meeting the performance requirements (e.g., robust and rapid dynamic configuration) being set for the autoconfiguration of MANETs.

After some time with no DRCP_ADVERTISE message, the client assumes that the DRCP server is not reachable anymore so it will move into the PRECONFIG state. If it has any preconfigured configuration information, the client will become a server for the corresponding link. Otherwise, the client moves to the SERVER_FIND state where it continues waiting for DRCP_ADVERTISE messages and sending DRCP_SOLICIT messages.

If a non-configured client receives a DRCP_ADVERTISE message, then it will go to the BINDING state. In the BINDING state it unicasts a DRCP_REQUES message to the source address of the DRCP_ADVERTISE message until it gets a DRCP_REPLY message. Once it receives a DRCP_REPLY message, the client moves to the BOUND state and cam immediately configure its interface with the received configuration information. There in no requirement for doing Duplicate Address Detection (DAD), as there is in DHCP, since the server does preemptive DAD. After being configured the client may periodically attempt to renew the lease by moving into the RENEWING state. It renews by a REQUEST-REPLY transaction. If it fails to get renewed, it moves to the SERVER_CHK state, where it attempts to find any DRCP server.

2.4.4 Basic DRCP Server Operation

There is no distinct DRCP module that separates the servers from the clients. Initially, all DRCP nodes function as clients. The configuration state on the corresponding Link and the configuration information being carried from the node, can acquire DRCP server privileges to the DRCP interface of this node that is directly connected on the Link. Specifically, if a DRCP node carries configuration information (including a pool of available IP addresses) for the non-configured interface under consideration, then it can become a DRCP server, in the case where there is not another DRCP server already configured on this Link. It can also become a server through receiving configuration information on its external management interface.

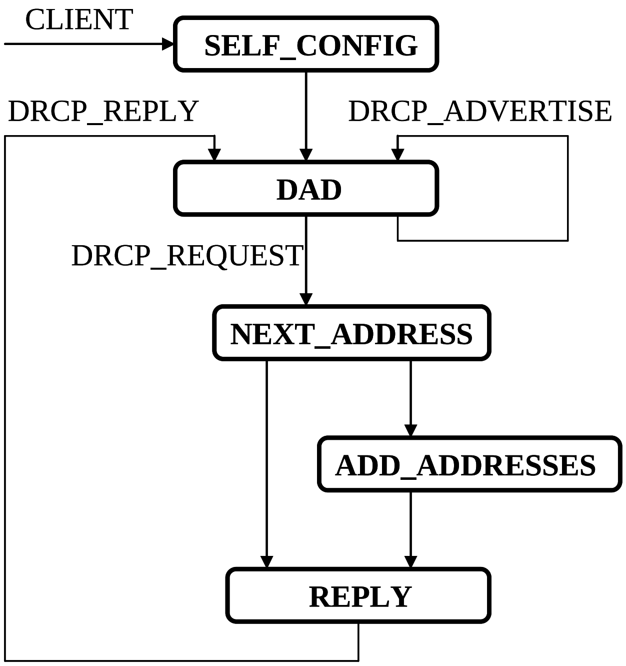


Figure 2.3. Simplified DRCP Server State Diagram

Figure 2.3 presents the functionality of a DRCP node once it becomes server. First, it enters the SELF_CONFIG state, where it assigns the first address from the local pool of available addresses to its own non configured interface. It then moves into the DAD state, where:

a) Listens on the DRCP_SERVER_PORT port

b) Checks the addresses to assign on the non-configured interfaces on the Link

c) Broadcasts or multicasts DRCP_ADVERTISE messages on the Link

The server performs preemptive DAD for addresses in its address pool, both for those it has leased as well as for those that are available for lease. This preemptive checking is done either by utilizing the Address Resolution Protocol (ARP) (for IPv4) or Neighboring Discovery (ND) (for IPv6). The checking of the leased addresses is also beneficial for the reclaiming of the unused such addresses.

Upon the reception of DRCP_REQUEST message, the server moves to the NEXT_ADDRESS state, where it associates the next available address with the client requesting it. Then, the server moves to the REPLY state and immediately sends a DRCP_REPLY message, where the configuration information intended for the client requested it (e.g., with a DRCP_REQUEST message), is included. In addition to the address assignment, the server also provides other configuration parameters, such as the location of default routers and network servers (i.e., DNS). The server does not expect an acknowledgment after sending the DRCP_REPLY message. The message will be retransmitted in the case where the server receives another DRCP_REQUEST message with the same transaction id.

Immediately, following the reception of a DRCP_REPLY message, the client can configure its interface. The application of DAD at the DRCP server has a threefold effect, the configuration time decreases, the broadcast/multicast messages to the clients are reduced and the client-to-client broadcast/multicast messages are eliminated. This elimination is highly desirable on some wireless links that are characterized from limited link broadcast. For example a roaming node's power level in CDMA2000 is set to reach its base station. This level may not be high enough for the node to reach the rest of the nodes. If the configuration server were placed at the base station, then it would be possible to broadcast from the server to all clients and from the client to the server but not from the client to the rest of the clients.

2.4.5 DRCP Message Format

The DRCP messages closely resemble those of DHCPv4; however the message size has been drastically reduced to preserve wireless bandwidth.

<table>
<tr><td>OP</td><td>VERS</td><td>HLEN</td><td>XID</td></tr>
<tr><td colspan="4">CLIENT_ID (8 bytes)</td></tr>
</table>

DRCP_REQUEST message

<table>
<tr><td>OP</td><td>VERS</td><td>HLEN</td><td>XID</td></tr>
<tr><td colspan="4">CLIENT_ID (8 bytes)</td></tr>
<tr><td colspan="4">IPv4 Address</td></tr>
</table>

DRCP_REPLY message

Figure 2.4. DRCP message format

Figure 2.4 shows, he DRCP_REQUEST and DRCP_REPLY message formats. In the case of IPv4, the basic DRCP message without options is 16 bytes, as opposed to the standard message format for DHCP without options, which totals to 236 bytes. Obviously, this results in substantial wireless bandwidth savings. Furthermore, if the DRCP server does not send periodic DRCP_ADVERTISE messages, then it can register a mobile host, provide it with a valid IP address, and configure it with the default router location in less than 100 bytes, which is half the size of a single DHCP message.

Depending on the speed and error rate of the wireless access networks, reducing the configuration overhead can be critical. The error rate can be important since larger messages result in higher probability of loss, which results in increased latency and bandwidth requirements. Clearly, it is undesirable to have per packet overhead; however, reducing the size and number of configuration messages can also significantly improve bandwidth efficiency for roaming users. DRCP minimizes configuration message size and the total number of messages.

2.4.6 Client Mobility

A node may require reconfiguration because it moves onto a new link or due to changes in other parts of the network. After detecting an external trigger, such as a layer 2 hand-off indication or a SNMPv3 request message, a node can reset its autoconfiguration process. Although it reduces the autoconfiguration protocol complexity, these external triggers may not always:

a) Be available

b) Be standardized

c) Have enough information (e.g., about whether it needs to reconfigure)

Moreover, completely resetting state may be inefficient. It may be useful, therefore, for the autoconfiguration protocol to determine the reconfiguration of a node using its own protocol. A key requirement for DRCP is configuration of roaming clients without requiring layer 2 support. DRCP does this mainly through the DRCP_ADVERTISE message.

If a configured DRCP client receives a DRCP_ADVERTISE message, it checks to determine the source id (by looking at the IP header). If the message has been originated from its configuration server, then it merely saves the current time as DRCP_TIME_LAST_ADVERTISE. If the message has been originated from a different server, then the client checks to determine if the new server is from an address on what it believes is its current link. If the DRCP_ADVERTISE comes from a server that it is not believed to be on the client's current link, then the client must perform new request-reply transaction with the new server (and return to the BINDING state).

2.5 Dynamic Configuration Distribution Protocol (DCDP)

The Dynamic and Rapid Configuration Protocol (DRCP) provides the mechanisms for the configuration of the nodes on a single link. The configuration information utilized from DRCP has to be preconfigured to the DRCP server of each link so that the corresponding link can be autoconfigured. Clearly, even though the DRCP protocol looks promising, does not have the ability to autoconfigure an entire network (e.g., a collection of large number of links and nodes).

Apart from DRCP, the efforts in the Internet community have so far focused on autoconfiguring hosts and small networks. The link configuration protocols, such as DRCP, PPP and DHCP still require server preconfiguration, which is geared towards environments where network dynamics are restricted to one hop at the network edge. DCDP has been proposed for the expungement of the latter limitation, expanding the functionality of the link configuration protocols, to larger and more dynamic networks.

The Dynamic Configuration Distribution Protocol's (DCDP) operational characteristics are orthogonal to those of the link configuration protocols. The latter focuses on the configuration of the nodes on a single link utilizing stored configuration information, as opposed to DCDP which is responsible for the management and distribution of the configuration information across a network of multiple links. DCDP has been designed independently of the link configuration protocols, so that it can be applied conjointly with anyone of these and expand their functionality to multiple links networks. By separating the distribution and management of configuration information, from the utilization of this information, we improve the robustness of the configuration approach. The dual protocol approach is attractive since it allows the improvement of each of the protocols separately and provides flexibility on the selection of the link configuration protocol.

DCDP does not require server-client preconfiguration among the DCDP nodes. They utilize the same software module and depending on the configuration information they own and the configuration state they are, can become from distributors (servers) to requestors (clients) and vice versa. Their communication module can be separated into three larger submodules, responsible for the exchange of intra-node and inter-node configuration information.

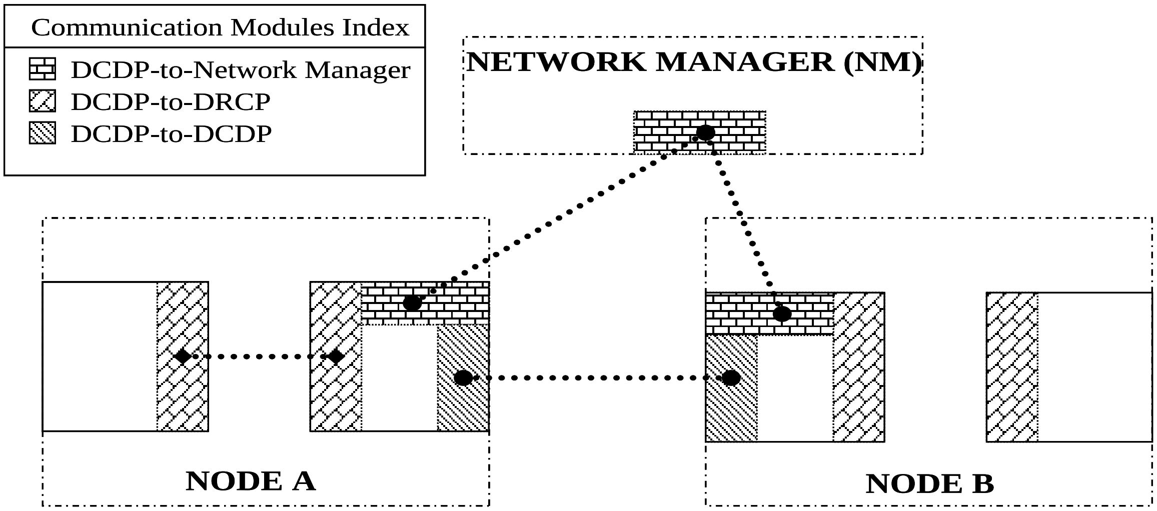


Figure 2.5. DCDP Communication Modules Diagram

These three submodules as they appear in figure 2.5 are:

a) The DCDP to DCDP submodule for the communication between different DCDP nodes

b) The DCDP to link configuration submodule (i.e., DRCP module), for providing the link configuration protocol with the appropriate configuration information required for the autoconfiguration of the links.

c) The DCDP to Network Manager submodule, which is utilized for bootstrapping and updating the DCDP protocol with the configuration information to be distributed across the network.

The DCDP functionality is presented in more detail in the following sections, where without loss of generality, we assumed that the underlying link configuration protocol is the DRCP. This assumption is useful to describe the specifics (e.g., format of messages and interaction with the link configuration protocol) of the protocol. This selection was done since we had already implemented DRCP, which fits better the MANETs compared to DHCP, PPP and SA. The implementation of DCDP was done with respect to DRCP module. Without loss of generality, the description of the DCDP messages and its functionality will be given with respect to this implementation. The application and interaction of DCDP with other link configuration protocols can happen in the same manner, with slight implementation modifications.

2.5.1 Basic DCDP-to-DRCP Communication

For the configuration of a link, DRCP requires preexisting configuration information. Without DCDP this information had to be stored in the DRCP node, so it could be utilized in case it was needed. With the addition of DCDP module, the configuration information is managed and distributed dynamically across the network, to be utilized from the DRCP modules that demand it for the link configuration.

When the configuration information reaches the DCDP node where the requestor DRCP module lies, then the DCDP module has to communicate with the DRCP module to transfer the appropriate configuration information. The basic message flow between the DCDP and DRCP modules is represented from figure 2.6.

Initially, DRCP requests (DRCP_POOL_REQUEST) configuration information (e.g., pool of available IP addresses, address of DNS) from the local DCDP process utilizing the DRCP_TO_DCDP_PORT port. If the DCDP module has configuration information available, it provides it to DRCP by sending a DRCP_POOL_REPLY message to the DCDP_TO_DRCP_PORT port. Otherwise, DCDP initiates the configuration information discovery phase by contacting neighboring DCDP modules, requesting configuration information. If this information is available, gets it and responds back to the local DRCP by sending a DRCP_POOL_REPLY message to the DCDP_TO_DRCP_PORT port, so that DRCP can proceed with the link configuration.

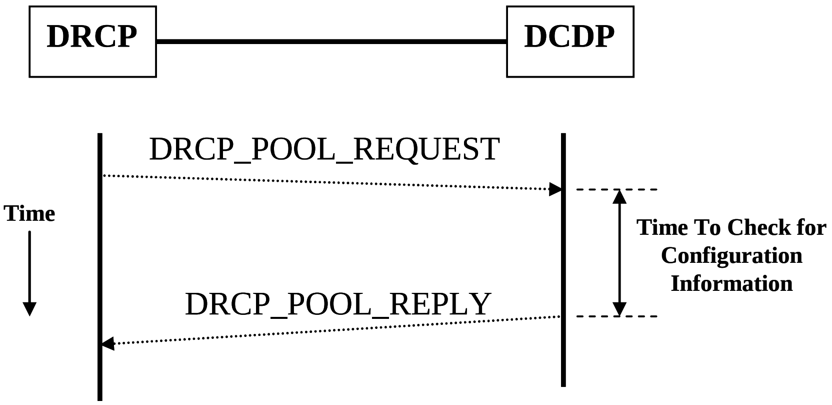


Figure 2.6. DCDP-to-DRCP Message Flow Diagram

The two messages that are responsible for the transfer of configuration information from DCDP to DRCP are:

- **DRCP_POOL_REQUEST**: When the DRCP module fails to get configured on its link, timeouts. A request for configuration information follows, by sending a DRCP_POOL_REQUEST message to its local DCDP process. The objective is to obtain and utilize configuration information to configure its own interfaces and become the DRCP server for the corresponding links, so that the rest of the non configured nodes can also be configured.
- **DRCP_POOL_REPLY**: This message is generated from the DCDP process and is in response to the DRCP_POOL_REQUEST message, received from the local

DRCP module. When the DCDP module succeeds on reserving configuration information for the DRCP module requesting it, transmits this message. The DRCP_POOL_REPLY message contains the configuration information that will be utilized from DRCP and is issued only in case of success. Otherwise the DCDP is not issuing any message to DRCP, which will eventually timeout.

Part of the configuration information that DRCP requests from DCDP is a pool of available IP addresses for the configuration of the nodes onto its link. The DCDP responds to DRCP by sending a subset of the set of available addresses that owns. The management and sharing of the pool of addresses is presented in more detail in section 5.5.

2.5.2 DCDP-to-DCDP Communication

The communication between DCDP and DRCP mainly happens via interprocess communication on the same node. There are cases where this may not be the case (e.g. when a node activates only one of the DCDP, DRCP modules), but these are special cases. The communication between DCDP modules happens always over the network and is more demanding in terms of designing. The communication between the DCDP modules is initiated when a DRCP module requests configuration information by sending a DRCP_POOL_REQUEST message to its local DCDP module. If the DCDP module has already obtained configuration information (e.g. pool of available addresses, addresses of routers and DNS), replies immediately to DRCP with a DRCP_POOL_REPLY message. Otherwise, DCDP has to search for available configuration information by querying its neighboring DCDP modules. The basic message flow diagram is presented on figure 2.7

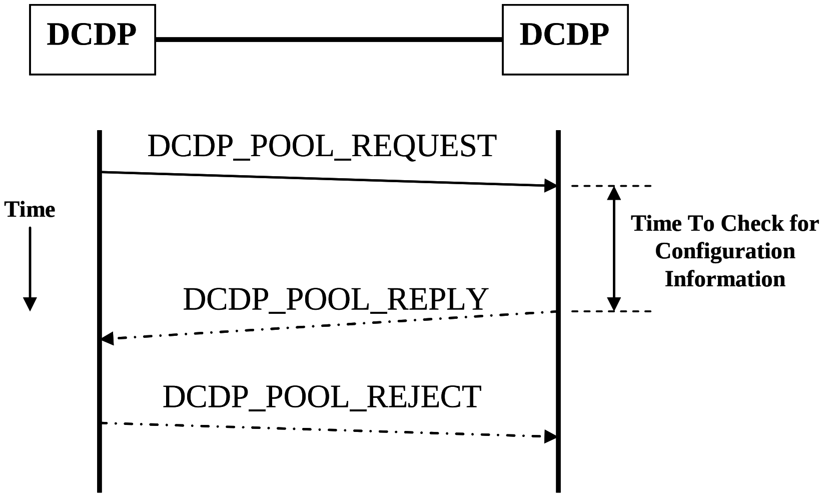


Figure 2.7. DCDP-to-DCDP Message Flow Diagram

The DCDP module is triggered to discover configuration information after receiving a DRCP_POOL_REQUEST message from the local DRCP process. DCDP initiates the discovery phase by broadcasting a DCDP_POOL_REQUEST message. The recipient DCDP modules of this message check the availability of configuration information. If there is available configuration information, they reply back by sending a DCDP_POOL_REPLY message. Otherwise they forward the request further to their neighboring DCDP modules. In the case where the requestor receives a DCDP_POOL_REPLY message and does not anymore requires the configuration offer, issues a DCDP_POOL_REJECT message to inform the remote DCDP. In the opposite case where the requestor does not receive any DCDP_POOL_REPLY message for a TIMEOUT time, timeouts and informs the local DRCP that there is not configuration information available.

The messages responsible for the DCDP to DCDP communication are:

- **DCDP_POOL_REQUEST**: The DCDP module broadcasts this message to its neighboring DCDP modules for discovering available configuration information. The transmission of this message is triggered from the reception of a DRCP_POOL_REQUEST message from the local DRCP process, or from the reception of a DCDP_POOL_REQUEST message from a neighboring DCDP module. In the latter case, the received DCDP_POOL_REQUEST message is forwarded further when there is not available configuration information to the local DCDP module.
- **DCDP_POOL_REPLY**: This message is issued from a DCDP module that had received a DCDP_POOL_REQUEST and it has available configuration information to be transferred to the requestor DCDP. The message contains the available configuration information (i.e. pool of available IP addresses, DNS address).
- **DCDP_POOL_REJECT**: When the DCDP requestor module receives a DCDP_POOL_REPLY message but the configuration information is not required anymore (e.g., will utilize earlier offer) issues a DCDP_POOL_REJECT message. The destination is the DCDP module offered the configuration information, so that it can recollect this information for future use. This message is important for the efficient utilization of the pool of available configuration information.

The DCDP-to-DCDP communication happens over UDP. Due to the dynamics of the environment under consideration, messages can be lost. For overcoming this problem, the configuration time and efficient utilization of the available configuration information have been traded off with the robustness of the approach. The approach is based on the combined utilization of negative acknowledgements and timeouts. When a DCDP module sends configuration information (i.e. pool of available addresses) it assumes that this information has been transferred to the requestor DCDP module, unless it receives a DRCP_POOL_REJECT message. The robustness of DCDP is improved compared to the case where the requestor DCDP had to acknowledge the acceptance of the offer. If DCDP was operating based on positive acknowledgment it would have been possible to offer addresses that already being used for the configuration of another part of the network. The latter could happen when the requestor DCDP receives the offer but its positive acknowledgment gets lost. By using negative acknowledgments, even if the acknowledgments get lost, the configuration information is assumed that has been accepted from the requestor DCDP and cannot be reused. The latter may not be true so we may end up utilizing inefficiently the available configuration information but the robustness of the protocol is more important, especially when we assume IPv6 (e.g. larger number of available IP addresses).

The DCDP module that offers the configuration information has to wait for time:

$$Time_{waiting} = \min\left(TIMEOUT, Time_{DCDP\_POOL\_REJECT}\right)$$

When most of the offers get accepted, the TIMEOUT waiting time dominates, so the configuration time of the network increases. The configuration information does not flow fast enough around the network due to the waiting time of the DCDP nodes to decide which of the configuration information they had offered can be reused and which of this information cannot.

2.5.3 DCDP-to-Network Manager Communication

The configuration information owned by a DCDP module is stored locally to the node in a file. This information is preconfigured to the node or it is obtained from other DCDP modules in response to DCDP_POOL_REQUEST messages. Obviously, the configuration information that flows into the network must have been preconfigured into at least one DCDP node.

What happens when the environment changes and the configuration information provided is not sufficient to configure appropriately the participating nodes? In that case the network configuration mechanism must be robust enough to allow new configuration information to be incorporated into the network from a Network Manager (NM). For that reason the DCDP module maintains the DCDP-to-Network Manager submodule through where new configuration information can flow into the DCDP module from a NM and from there can be distributed across the network via the DCDP-to-DCDP submodule.

2.5.4 State Flow Diagrams and Messages Format

The DCDP description we provided in earlier sections reveals the principal operation of DCDP module. In this section a more detailed description of the module is given, via the use of state flow diagram. The DCDP state flow diagram has been split into two parts. The one part describes the sequence of actions related to the interaction of DCDP module with its local DRCP module and the other part is dedicated to the interaction between cooperating DCDP modules

In both cases the DCDP module is initialized by checking (CHECK_INFO) the quantity of the configuration information that has available either for self utilization or for distribution. Once it has gone through this initialization and depending on the availability or not of configuration information the DCDP module resides in the HAVE_INFO or DO_NOT_HAVE_INFO state, respectively; until the arrival of a message. This message can have three possible sources of originations:

- The local DRCP module
- A Neighboring DCDP module
- The Network Manager

When the message has been originated from the DRCP module, the functionality of DCDP is represented from the following state flow diagram in figure 2.8. Currently the only message that triggers some action in DCDP is the DRCP_POOL_REQUEST. With this message DRCP requires configuration information to configure its interfaces and the nodes residing onto the corresponding links. Upon the reception of this message, DCDP reacts appropriately depending on its current state (HAVE_INFO or DO_NOT_HAVE_INFO). If DCDP is in the HAVE_INFO state then checks (AVAILABILITY_DRCP) if the configuration information available suffices to fulfill the DRCP request. If the configuration information is adequate, DCDP enters the SEND_INFO state and a DRCP_POOL_REPLY is transmitted to local DRCP, which contains the configuration information offer. Following that, DCDP checks its configuration information repositories (CHECK_INFO) and returns to HAVE_INFO or DO_NOT_HAVE_INFO states. In the case where DCDP is in the DO_NOT_HAVE_INFO state upon the reception of a DRCP_POOL_REQUEST message, then the configuration information discovery phase is initiated. This is done by transmitting a DCDP_POOL_REQUEST message to its DCDP neighbors. The DCDP

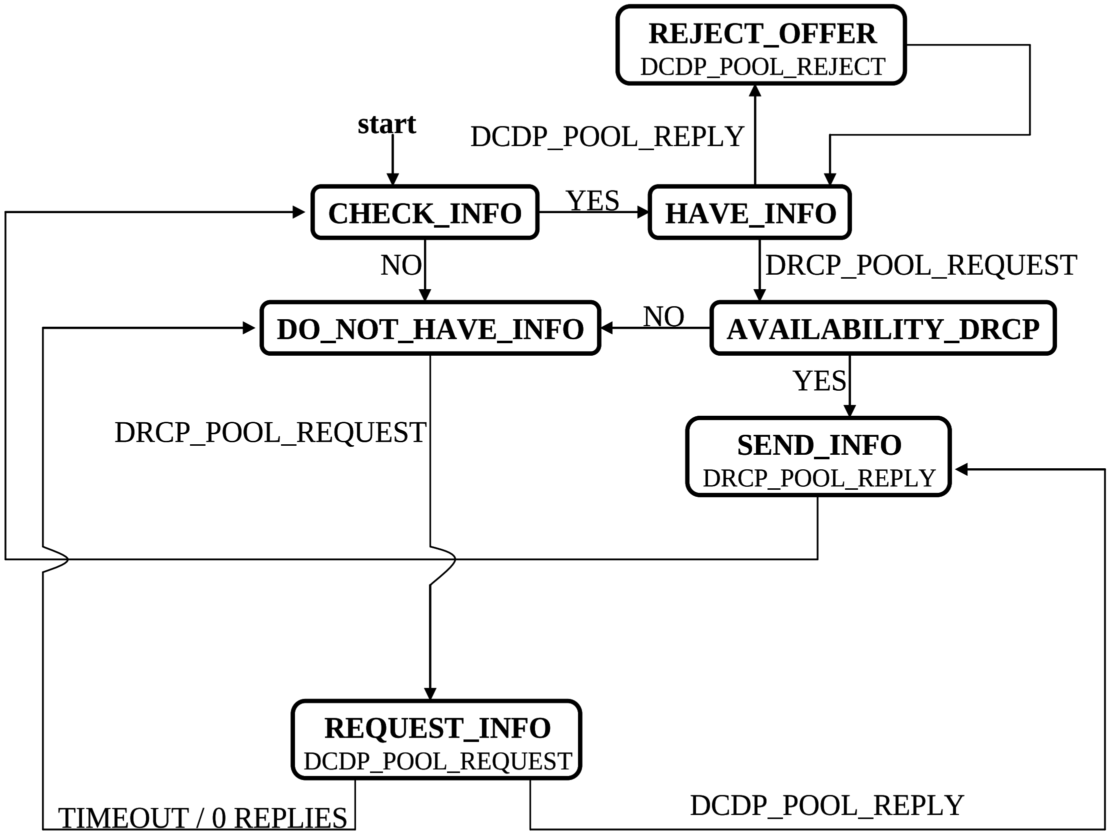


Figure 2.8. DCDP state flow diagram (interaction with local DRCP)

waits for time:

$$Time_{waiting} = \min\left(TIMEOUT, Time_{DCDP\_POOL\_REPLY}\right)$$

before its next action. If there is not any configuration information offers (e.g. there is not any DCDP_POOL_REPLY message) then the returns to the DO_NOT_HAVE_INFO state without replying to the DRCP module. The DRCP module will eventually timeout and will have to reenter the DRCP server discovery phase. In case of the reception of DCDP_POOL_REPLY message then the DCDP updates its configuration information repositories and replies to DRCP request (SEND_INFO) by sending a DRCP_POOL_REPLY message. After this reply, checks (CHECK_INFO) the amount of configuration information available and returns to the HAVE_INFO or DO_NOT_HAVE_INFO states respectively. In the case that more than one replies arrive at DCDP in response to the DCDP_POOL_REQUEST message, then these offers are getting rejected (REJECT_OFFER) by transmitting a DCDP_POOL_REJECT message to the source of the offer.

When a DCDP module receives a message originated from another DCDP module, this event triggers the DCDP interaction functions, which are described from the following figure 2.9. Similarly, the DCDP module after checking (CHECK_INFO) its configuration information repositories lies in HAVE_INFO or DO_NOT_HAVE_INFO state. Depending on which of these two states it is, when it receives a DCDP_POOL_REQUEST message, DCDP acts accordingly. If it is in HAVE_INFO state, DCDP checks (AVAILABILITY_DCDP) if the available amount of configuration information suffices to provide an offer. In the case where the configuration information is enough for an offer, then enters the SEND_INFO state where it constructs and transmits a DCDP_POOL_REPLY message destined to the requestor DCDP module. Following this action, the DCDP rechecks its configuration information repositories (CHECK_INFO) and depending on the availability resides on the HAVE_INFO or the NOT_HAVE_INFO state.

In the case where the DCDP module was on the HAVE_INFO state but the available configuration info is not enough for providing an offer then the module enters the REQUEST_INFO state.

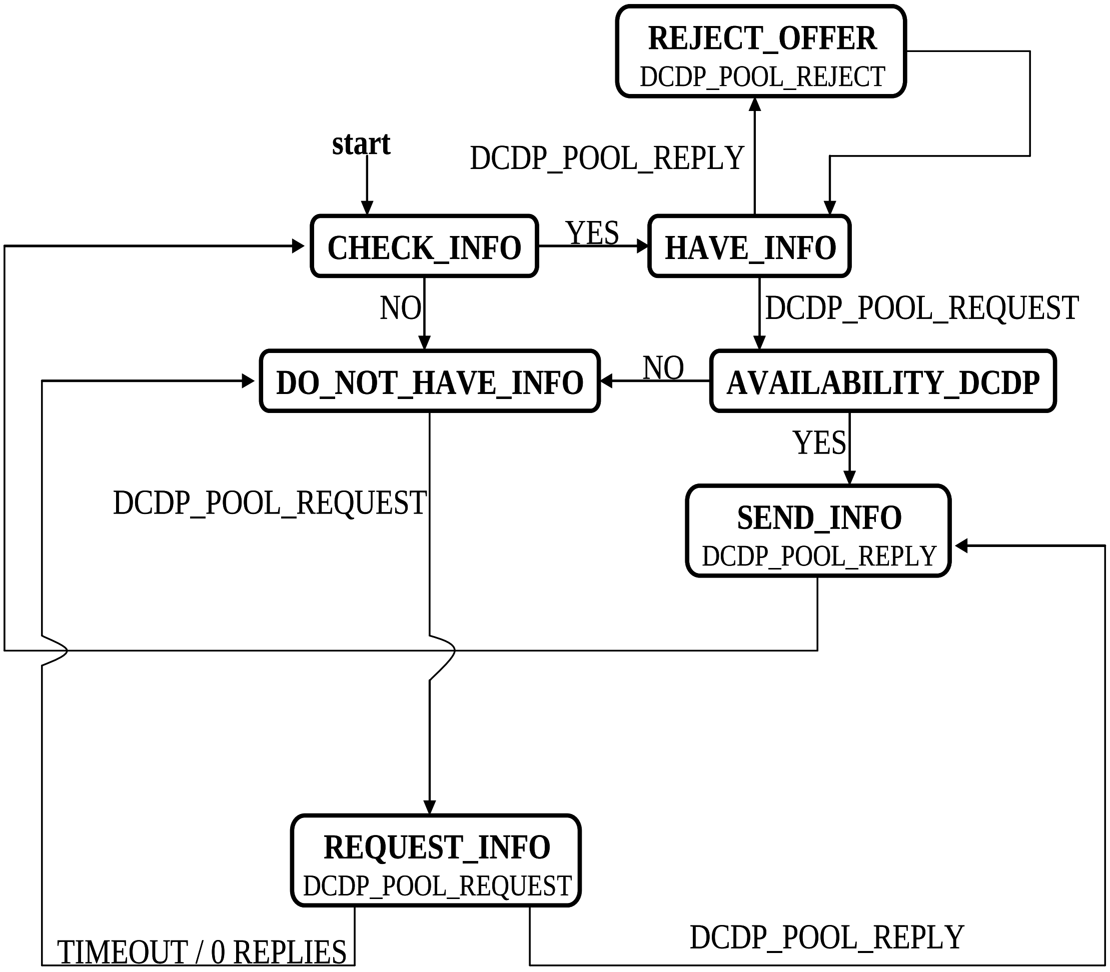


Figure 2.9. DCDP state flow diagram (interaction with DCDP)

Upon entering this state, DCDP initiates the configuration information discovery phase by transmitting a DCDP_POOL_REQUEST message. The module remains in this state for time:

$$Time_{waiting} = \min\left(TIMEOUT, Time_{DCDP\_POOL\_REPLY}\right)$$

If there are no configuration information offers then the module enters the DO_NOT_HAVE_INFO state and does not respond to the DCDP_POOL_REQUEST message that originally triggered the discovery phase. In the case where there is an offer then it constructs a DCDP_POOL_REPLY message and sends it (SEND_INFO) to the requestor DCDP process as a response to the previously received DCDP_POOL_REQUEST message. As before the DCDP module checks (CHECK_INFO) the amount of the remaining available configuration information and resides in the HAVE_INFO or the DO_NOT_HAVE_INFO state.

When DCDP module requests configuration information then multiple offers can be received. If this happens, then the DCDP module enters the REJECT_OFFER state. It transmits a DCDP_POOL_REJECT message to inform the provider of the offer about its decision not to utilize the offered configuration information.

The scenario remaining to be explored is when the DCDP module receives a DCDP_POOL_REQUEST message but does not have any configuration information (DO_NOT_HAVE_INFO state). The reaction to a DCDP_POOL_REQUEST message is the same as before, when there is not enough available configuration information, so a discovery phase is initiated.

The format of the messages that are exchanged during the DCDP-DRCP or DCDP-DCDP interactions is similar to the format of the DRCP messages described earlier. These messages are lightweight aiming on the overhead reduction, which is critical for the characteristics of the network environments under consideration (MANETs). Figure 2.10 shows the standard DCDP header (with sizes of fields shown in bytes).

<table>
<tr><td>Ver</td><td>OP</td><td>- (1)</td><td>hlen (1)</td><td>CP (1)</td></tr>
<tr><td colspan="5">ID (8)</td></tr>
<tr><td colspan="5">Body (variable)</td></tr>
</table>

Figure 2.10. DCDP Header Format

The fields are defined as follows:

- **Ver:** Version number
- **OP:** Indicates DCDP message type (i.e., DCDP_POOL_REPLY)
- **hlen:** Length ofbss header in 4 byte words
- **CP:** Configration Information Priority

  If CP < 0 private (e.g., 10.x.x.x)

  If CP > 0 globally unique (e.g. 112.4.1.61)
- **ID:** 64 bit configuration identifier
- **Body:** message (i.e., for a DCDP_POOL_REPLY message this segment contains the offer of the available IP addresses)

2.5.5 Pool of Available Addresses Management

There are two types of requests for available configuration information. These two types are classified with respect to their originator module (e.g. DRCP or DCDP). Important part of the configuration information being requested is a pool of available addresses. The way the pool of available addresses is managed is very important for their efficient distribution throughout the network. The importance of the management of available addresses is expressed from the following two lemmas:

Lemma 2.1: If the pool $\mathbb{S}$ of available address has cardinality $|\mathbb{S}|$, the network $\mathbb{N}$ to be configured has size $|\mathbb{N}|$ and the relation between $|\mathbb{S}|$ and $|\mathbb{N}|$ is:

$$O(|\mathbb{S}|) = O(|\mathbb{N}|)$$

then the distribution of the available addresses has to resemble the topology of the network, so that the distribution is efficient and all the nodes are configured.

Lemma 2.2: If the network considered is an IP network and the available addresses to be distributed are IP addresses, then the interfaces of each link has to be configured with IP addresses of the same netmask, so that the routing table's $\mathbb{R}$ size $|\mathbb{R}|$ will not grow linearly with the network size.

The management of available addresses in the current distribution of DCDP is described in this section. The management depends on the request received from the DCDP module. There are two types of requests, the DRCP_POOL_REQUEST (from the local DRCP module) and the DCDP_POOL_REQUEST (from a neighboring DCDP module). Since we considered IP networks, the management has been customized for IP addresses and aims on satisfying the conditions imposed from the above mentioned lemmas (2.1 and 2.2).

When the DCDP module has a pool of available IP addresses and receives a DRCP request for offering configuration information, the reply consists of a subset of the owned pool of available IP addresses. The offered pool *V* has two characteristics:

- It is continuous

- It has size $\left|V\right| = 2^n, n \in \mathbb{Z}_+^*$

These properties satisfy the condition imposed by Lemma 2.2. The addresses offered to the local DRCP module will be utilized for the configuration of the local interfaces and of the network entities that belong on the same link. Due to the functionality of DRCP in terms of assigning addresses and the properties of the offered pool, the addresses assigned on the network entities of the same link can be represented by a single netmask. This reduces significantly the size of the IP routing tables.

Currently, the offer from the DCDP to DRCP module has size of 256 addresses and is of the form X.X.X.0 to X.X.X.255 so that the netmask representing this link is 255.255.255.0. The benefit of selecting this type of offer is obvious when we consider the two scenarios where in the first one the offer is 10.1.2.0 to 10.1.2.255 and in the second one is 10.1.1.128 to 10.1.2.127. In the first scenario the addresses can be represented from a single netmask (e.g. 255.255.255.0). The addresses of the second offer require two different netmasks for their representation (e.g., 255.255.255.128 for the 10.1.1.128 network and 255.255.255.0 for the 10.1.1.2.0 network). Since the nodes that belong on the same link, appear to be on different networks, require distinctive representation on the routing table, which unavoidably will increase the size of the routing tables carried from the individual routers across the network.

The DCDP module response to a request for an offer originated by another DCDP module (DCDP_POOL_REQUEST) differs from the corresponding response originated from the local DRCP module. The size of the pool V_{DCDP} of available addresses offered to a DCDP module is larger than the size of the pool V_{DRCP} offered to DRCP. The relation of the cardinalities between the two sets of addresses is:

$$\left|V_{DCDP}\right| = K\left|V_{DRCP}\right|, K \in \mathbb{Z}_{+}^{*} \qquad (2.1)$$

The DCDP to DCDP offer involves larger pool of addresses because this pool will be used from the DCDP module to respond to the local DRCP's request for an offer so it is required that $\left|V_{DCDP}\right| \geq \left|V_{DRCP}\right|$. The relation (2.1) can be justified further because it is possible the same DCDP module to respond to requests from multiple DRCP modules, since the same DCDP module can be responsible for multiple DRCP processes (e.g. multiple links).

The justification of (2.1) is straight forward except from the precise definition of the parameter K. In the optimal scenario, the selection of the parameter K is different for each DCDP module and depends on the number of DRCP server processes for which the DCDP module is responsible. If K^* is the optimal selection for K for a single DCDP module then the effects due to its suboptimal selection are:

- If $K > K^*$ then this will result in the inefficient utilization of the global pool of available addresses, since many of the addresses offered will not be requested from DRCP server processes, but also will not be available to parts of the network that demand them. This effect is more severe when the global pool of addresses has size on the order of the network size.
- If $K < K^*$ then the configuration time and overhead will increase significantly, since DCDP will not be adequate to satisfy the all the DRCP requests. This will result on extra DCDP to DCDP requests for available addresses so that the local DRCP requests can be satisfied. The extra DCDP to DCDP requests (DCDP_POOL_REQUEST) will increase the overhead imposed from the

configuration process and the configuration time will increase due to the time required for the extra requests to get served.

The optimal selection of K requires both knowledge of the topology of the network and the status (server or client) of the DRCP processes. Since neither of this information is known while the network configuration is taking place, it is impossible to specify the optimal parameter value K^*.

Due to the lack of K^* knowledge, the DCDP adopts a suboptimal heuristic for determining the size of the pool to be offered as a response to a DCDP_POOL_REQUEST. The heuristic is based on splitting the available pool of addresses into two equal size subsets. The one subset is retained from the DCDP module for its future needs and the second subset constitutes the pool to be offered. Due to the requirements of the DCDP to DRCP offers, the splitting has to be done in such a way that the resulting subsets have size as similar as possible and these requirements are met. As it was mentioned above, when IP addresses are considered, the latter requirements are best satisfied when the DCDP to DRCP offers are of the form X.X.X.0 to X.X.X.255. The combination of the DCDP to DRCP offer with (2.1) and the heuristic of splitting the available pool into two subsets of equal size, provides the general guidelines of the splitting method for generating the DCDP to DCDP offer (DCDP_POOL_REPLY). Along these guidelines the general form of the pool to be offered is:

$$A''.B''.C''.0 - D.E.F.255 \tag{2.2}$$

where $A'', B'', C'', D, E, F \in [0, 255]$ and $A'', B'', C'', D, E, F \in \mathbb{Z}$

The heuristic method followed by the DCDP module to construct an offer of this type which also satisfies the imposed requirements can de described by assuming the existence of an available pool of the form:

$$A.B.C.0 - D.E.F.255 \tag{2.3}$$

where $A, B, C, D, E, F \in [0, 255]$ and $A, B, C, D, E, F \in \mathbb{Z}$

When this DCDP module receives a DCDP_POOL_REQUEST message, checks if there is available configuration information to be offered for the fulfillment of the request. Part of the available configuration information is the pool of available addresses. We assume that the DCDP module has available configuration information included the pool (2.3). For the construction of the DCDP_POOL_REPLY, the DCDP module has to split the available pool of addresses into two subsets of almost equal size, and of the following form:

$$A.B.C.0 - A'.B'.C'.255 \text{ and } A''.B''.C''.0 - D.E.F.255 \tag{2.4}$$

where

$A, B, C, A', B', C', A'', B'', C'', D, E, F \in [0, 255]$ and $A, B, C, A', B', C', A'', B'', C'', D, E, F \in \mathbb{Z}$

The DCDP module has to determine the values of $A', B', C', A'', B'', C''$, so that it can determine the pool offer. The computation for each of these parameters is done as follows and is based on the knowledge of the parameters A, B, C, D, E, F, which determine the owned pool.

Step I: Determine the size of the pool

The computation is based on the following base-256 subtraction:

$$|\mathbb{S}| = (D - A) \times 256^3 + (E - B) \times 256^2 + (F - C) \times 256 + 255$$

Step II: Determine the size of each resulting subset of the pool

The size of each one of the two resulting pools of IP addresses is:

$$\left|\mathbb{S}_1\right| = \left\lfloor \frac{\left|\mathbb{S}\right|}{2} \right\rfloor \text{ and } \left|\mathbb{S}_2\right| = \left|\mathbb{S}\right| - \left|\mathbb{S}_1\right|$$

Step III: Convert the sizes of the resulting subsets into base-256 numbers

Use base-256 arithmetic conversion to convert the base-10 sizes ($\left|\mathbb{S}_1\right|, \left|\mathbb{S}_2\right|$) of the resulting pools ($\mathbb{S}_1, \mathbb{S}_2$) into base-256 numbers:

$$\left|\mathbb{S}_1\right|_{256} = K_3 K_2 K_1 K_0 \text{ and } \left|\mathbb{S}_2\right|_{256} = K_7 K_6 K_5 K_4$$

where

$$K_7, K_6, K_5, K_4, K_3, K_2, K_1, K_0 \in \left[0, 255\right] \text{ and } K_7, K_6, K_5, K_4, K_3, K_2, K_1, K_0 \in \mathbb{Z}$$

If $K_0 \neq 255$ then go to step IV, else continue to step V.

Step IV: Adjust the sizes (if required)

Since $K_0 \neq 255$ we have to adjust the sizes of the pools defined at step II, so that $K_0 = 255$. This is done by adding the appropriate number to K_0 and subtracting the same number from $\left|\mathbb{S}_2\right|$.

$$O_{ffset} = 255 - K_0 \Rightarrow K_0' = 255 \text{ and } \left|\mathbb{S}_2'\right| = \left|\mathbb{S}_2\right| - O_{ffset} \Rightarrow K_4' = 0$$

After the adjustment the base-256 size of each one of the two pools is:

$$\left|\mathbb{S}_1'\right|_{256} = K_3 K_2 K_1 255 \text{ and } \left|\mathbb{S}_2'\right|_{256} = K_6 K_5 K_4 0$$

where $K_6, K_5, K_4, K_3, K_2, K_1 \in \left[0, 255\right]$ and $K_6, K_5, K_4, K_3, K_2, K_1 \in \mathbb{Z}$

Step V: Determine the exact subsets of the initial pool

Having determined the size of the subsets of the initial pool

$$\left|\mathbb{S}_1\right|_{256} = K_3K_2K_1 255 \text{ and } \left|\mathbb{S}_2\right|_{256} = K_6K_5K_4 0$$

where $K_6, K_5, K_4, K_3, K_2, K_1 \in [0, 255]$ and $K_6, K_5, K_4, K_3, K_2, K_1 \in \mathbb{Z}$,

the two resulting pools can be exactly described:

$$\begin{aligned}\mathbb{S}_1 &= \left[\left(A.B.C.0\right)_{256} ... \left[\left(A.B.C.0\right)_{256} + \left|\mathbb{S}_1\right|_{256}\right]\right] = \\ &= \left[\left(A.B.C.0\right)_{256} ... (A'.B'.C'.255)_{256}\right]\end{aligned}$$

$$\begin{aligned}\mathbb{S}_2 &= \left[\left[\left(A'.B'.C'.255\right) + \left(1\right)_{256}\right] ... \left[\left(A'.B'.C'.255\right) + \left(1\right)_{256} + \left|\mathbb{S}_2\right|_{256}\right]\right] = \\ &= \left[\left(A''.B''.C''.0\right)_{256} ... \left(D.E.F.255\right)_{256}\right]\end{aligned}$$

The $\mathbb{S}_2$ pool will be included in the DCDP_POOL_REPLY message and the $\mathbb{S}_1$ will be stored for the future needs of the local DCDP module.

The format of the offered pools is justified from the requirements imposed from the lemmas 2.1 and 2.2 for the efficient utilization of the available IP addresses and the minimization of the size of the routing tables. What has not been justified yet is the selection of splitting the initial pool into subsets of equal size. Since the optimal K^* in (2.1) cannot be determined in real time, then a suboptimal *K* has to be selected. By offering half of the owned pool as a response to a DCDP_POOL_REQUEST message, a suboptimal value for *K* is indirectly selected. This value is the optimal in the special case where the network grows symmetrically (e.g., in a tree fashion). When the network topology does not grow symmetrically, the effect of the suboptimality of *K* will be noticeable on the time of the configuration process and not on the efficient utilization of the available addresses. The addresses will still be utilized efficiently due to the capability of DCDP module to request configuration information from neighboring DCDP modules whenever this information is needed and is not available locally. If a DCDP module offers the largest part of its initial pool to satisfy DCDP_POOL_REQUEST messages and there is not available configuration information to satisfy its own needs, then it will issue a DCDP_POOL_REQUEST message, asking the neighboring DCPD modules for available configuration information. The DCDP_POOL_REPLY message that will be received may contain part of the pool that initially was owned from this DCDP module or a new pool. By acting in that fashion the available addresses are transferred to the parts of the network where are needed for the configuration of the network entities. Obviously, the efficiency of address utilization does not depend much on the selection of *K* but mostly depends on the format of the offer, which has to match the format of the DCDP to DRCP offers. As we mentioned, the configuration delay is the only aspect of the process that will be affected from the selection of *K*. The delay imposed from the selection of *K* is strongly correlated with the topology of the network and the order the various network entities require configuration.

The above description of the splitting of the initial pool it is based on the fact that the pool is of the form:

$$A.B.C.0 - D.E.F.255$$

where $A, B, C, D, E, F \in [0, 255]$ and $A, B, C, D, E, F \in \mathbb{Z}$

If the pool is not of this form then the heuristic method of splitting cannot produce subsets of the form imposed by the lemmas 2.1 and 2.2. For that reason, some modifications have to be made so that the heuristic can be applied independently of the form of the initial pool. The modification was applied on the description of the pool instead on the splitting heuristic. This is because the heuristic satisfies the requirements for the efficient utilization of the available addresses and the minimization of the routing tables. The modification applied on the description of the pool can be described from the adjustment of its size so that it can be described of the form (2.3). To achieve this, the limits of the pool have to be extended appropriately, so that the new pool is larger and follows the format (2.3). If the initial pool is of the general form:

$$A.B.C.L - D.E.F.M$$

where $$\begin{gathered} A,B,C,D,E,F \in [0,255] \text{ and } A,B,C,D,E,F,L,M \in \mathbb{Z} \\ \left(L \in (0,255]\right) \parallel \left(M \in [0,255)\right) \end{gathered}$$

then we subtract from L the appropriate offset so that it becomes 0 and we add to M the appropriate offset so that it becomes 255.

$$O^1_{ffset} = L \text{ and } O^2_{ffset} = 255 - M$$

Since we have added to the pool, addresses that cannot be used, we have to mark these addresses as unusable.

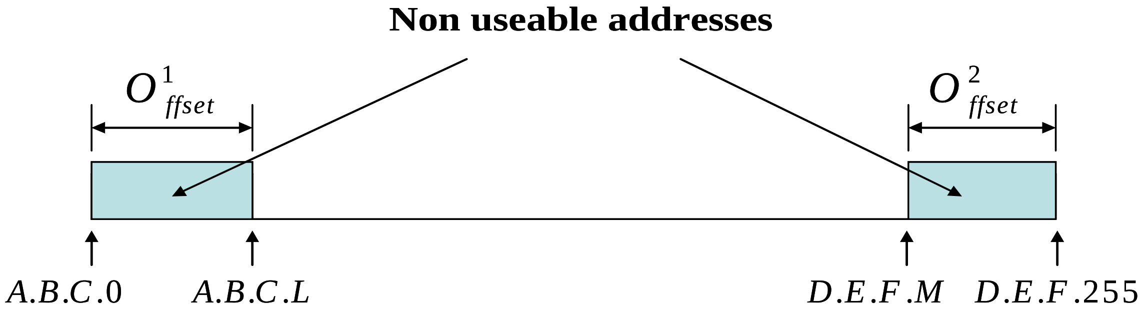


Figure 2.11. Adjustment of irregular initial pool

Even though these addresses will be offered, they could never be utilized for the configuration of the network entities, but the heuristic can be applied without any modification and will satisfy the splitting requirements.

2.6 Overview of the Complete IP Autoconfiguration Suite

The DRCP and DCDP modules constitute the core of the autoconfiguration suite. The complete autoconfiguration suite consists of more modules that are of equivalent importance for the successful application of the autoconfiguration process. Apart from the DRCP and DCDP modules, the developed suite consists of the following modules, which will briefly describe in this section:

- Adaptive Configuration Agent (ACA)
- Yelp Announcement Protocol (YAP)
- Configuration Information Database

Figure 2.12 shows the IPAS components that instantiate each of the main functions of a complete autoconfiguration suite. The Dynamic Configuration Distribution Protocol (**DCDP**) and Dynamic and Rapid Configuration Protocol (**DRCP**) perform the configuration distribution. The Configuration Database stores the configuration and network information reported by the Update Protocol (**YAP**). And, the Adaptive Configuration Agent (**ACA**) is the "brains" in the configuration process.

The configuration process can be pictured as a closed feedback loop. The ACA distributes new configuration through DCDP to nodes in each subnet. DRCP configures the interfaces within a subnet. Interfaces configured by DRCP, report configuration information and nodes capabilities to the configuration server via the

YAP protocol. The configuration server stores this information in the Configuration Database. To complete the cycle, the ACA node contacts the Configuration Database locally or remotely to get the latest configuration information. After processing this configuration information, the ACA may decide to reconfigure the network and distribute new configuration information, starting the cycle once more.

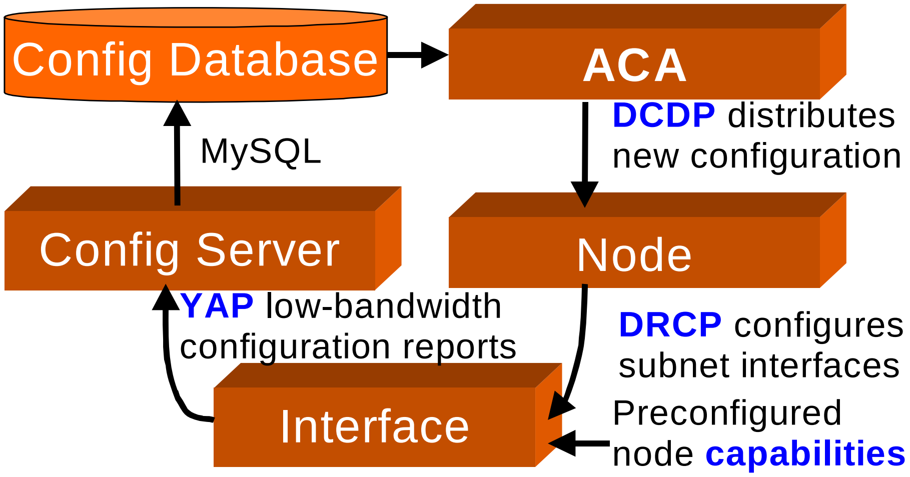

Config Database
ACA
DCDP distributes new configuration
MySQL
Config Server
Node
YAP low-bandwidth configuration reports
DRCP configures subnet interfaces
Interface
Preconfigured node capabilities

Figure 2.12. IPAS Components

The following paragraphs explain in more detail what are the functionalities of each of the IPAS modules and how they interoperate (a short description of DRCP and DCDP modules is included).

Dynamic Configuration Distribution Protocol (DCDP)

At the heart of IPAS (see Figure 2.12) is the Dynamic Configuration Distribution Protocol (**DCDP**). DCDP is a robust, scalable, low-overhead, lightweight (minimal state) protocol designed to distribute configuration information on address-pools and other IP configuration information (e.g., DNS Server's IP address, security keys, or routing protocol). Designed for dynamic wireless battlefield, it operates without central coordination or periodic messages. Moreover, DCDP does not rely on a routing protocol to distribute information.

Dynamic Registration and Configuration Protocol (DRCP)

DCDP relies on the Dynamic and Rapid Configuration Protocol (**DRCP**) to actually configure the interfaces. DRCP borrows heavily from DHCP, but adds features critical to roaming users. DRCP can automatically detect the need to reconfigure (e.g., due to node mobility) through periodic advertisements. In addition, DRCP allows for: a) efficient use of scarce wireless bandwidth, b) dynamic addition or deletion of address pools for supporting server fail over, c) message exchange without broadcast, and d) clients to be routers.

Yelp Announcement Protocol (YAP)

The Configuration Database Update Protocol (**YAP**) is a simple bandwidth efficient reporting mechanism for dynamic networks. YAP has three elements: 1) Clients running on every node periodically report its node’s capabilities, configuration, and operational status, 2) Relays forwarding information from clients to a server, and 3) Server storing the information in a configuration database (see Figure 2.13). The *capabilities* say, for example: “This node can be a DNS server with priority 0” or “a YAP server with priority 3” (priority reflecting a node’s willingness to perform a function). Other YAP information includes name and IP address, Rx/Tx packets, bit rate, link quality, routing table, and address pool.

Configuration Information Database

The Configuration Information Database can be centralized or distributed depending on the type and requirements of the network under consideration. It is used for storing and accessing information related to the configuration status of the network, and of its participating network entities. The database is accessed from YAP for storing the collected configuration information and from ACA for obtaining future configuration decisions. The configuration information stored in this database, is the information collected from YAP and includes but is not limited to the capabilities of the participating network entities, their names and IP addresses, the number of Rx/Tx packets, the bit rate, the link quality, the routing tables and the pools of available addresses.

Adaptive Configuration Agent (ACA)

The brain of IPAS is the Adaptive Configuration Agent (ACA). The ACA can even reset the network and distribute a new address pool from human input or from a predefined private address pool (e.g., 10.x.x.x). The configuration decisions are distributed to the network's nodes through the DCDP process. Through the Configuration Database (filled by YAP), ACA observes the state of the network, which allows it to initiate reconfiguration based on its rules or policies. The rules in the ACA are specific to the mission and network characteristics. Currently, ACA has a few simple and general rules, such as selecting a new DNS server if the current one does not report.

Interoperation of IPAS Modules

In each subnet there is at least one DRCP server responsible to configure interfaces. The rest of nodes in the subnet may perform as DRCP clients. The ACA runs on a dedicated node in the network. The YAP server, the ACA and the Configuration Database can either be located in the same node or be distributed. However, for load balancing and fault tolerance reasons those entities is better to be distributed, when the network environment under consideration is dynamic.

In each node, independently of its capabilities, there are three processes running:

- DCDP
- DRCP (server or client)
- YAP (client, relay or server)

Figure 2.13 shows how these processes communicate on a single node and how interoperate in different nodes in the network. If the node is the ACA, then its ACA process passes information to DCDP. The DCDP processes communicate with other nodes to distribute configuration information. At each DRCP server, the DCDP process passes configuration information to the DRCP process, so this can configure interfaces in the corresponding subnet. If there is a change or update in configuration information the DRCP process informs the local YAP client, which sends the information to the YAP relay for the subnet, which in turn relays the information to the YAP server. If the node is the YAP server it collects the information and stores it either locally or remotely on a configuration database. The ACA node can contact the Configuration Database locally or remotely to get the latest configuration information.

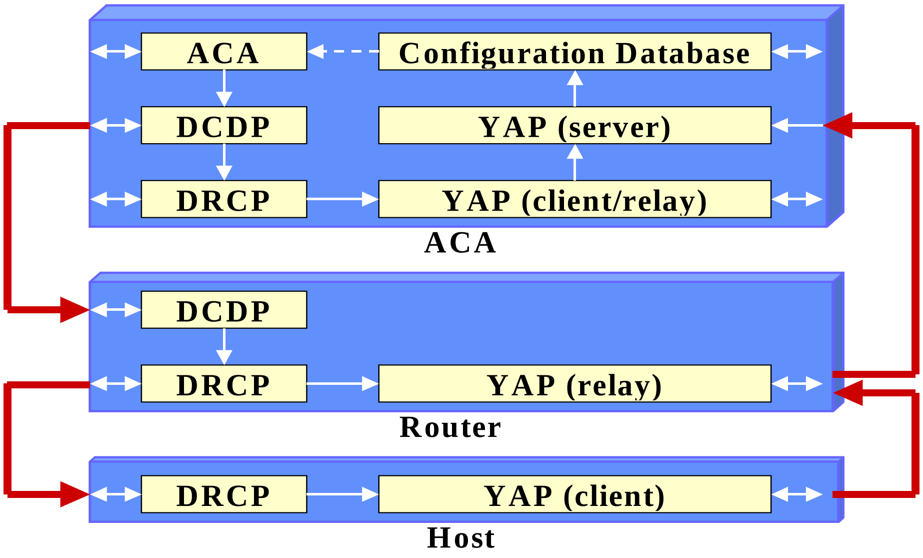


Figure 2.13. IPAS inter-process and inter-node communication

2.7 Implementation Based Performance Analysis

This section presents large scale performance results based on the implementation of the autoconfiguration algorithms and protocols in a small testbed. To study the feasibility of the approach to network configuration, IPAS modules were implemented in a lab testbed. Five Linux laptops were connected, each equipped with two 802.11 cards, as shown in figure 2.14 and a scenario to validate the capabilities of IPAS was set up. IP address configuration, dynamic routing protocol configuration and dynamic hierarchy configuration was presented. Node “**a**” performs the ACA function, initiating the distribution of IP addresses (initial pool was 192.1.1.1-192.1.38.255) and other configuration information. Figure 2.14 shows how the interfaces and subnets get configured and figure 2.15 demonstrates the configuration information flows between the IPAS modules. The DCDP modules distribute the pool of addresses and the DRCP modules utilize the pool of addresses obtained from DCDP to configure interfaces in their subnet.

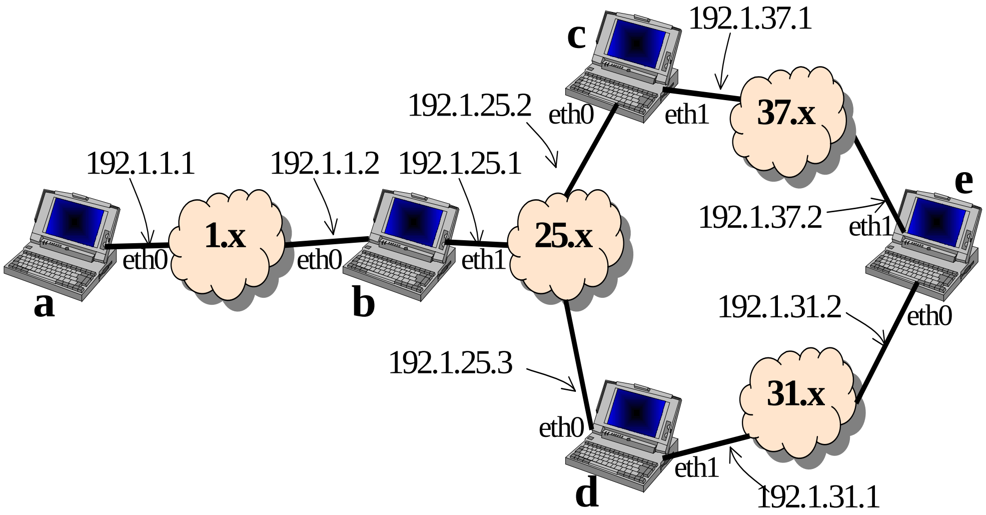


Figure 2.14. Network Autoconfiguration Testbed and IPAS Message Flow

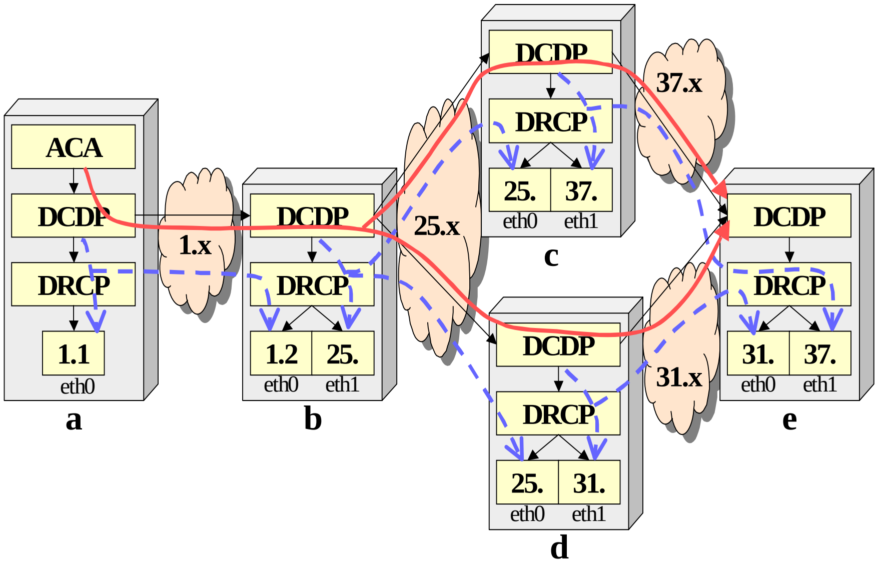


Figure 2.15. IPAS Message Flow

Based on experimental and simulation results, figure 2.16 shows the IPAS configuration time and overhead as a function of the number of nodes in the network.

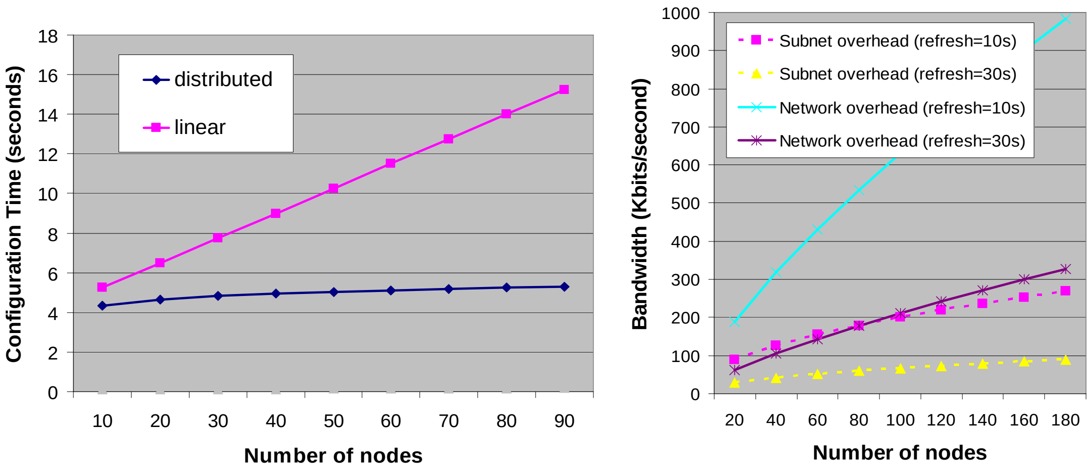


Figure 2.16. IPAS Configuration Time and aggregate overhead

The above figure shows the configuration time (primarily due to DCDP) for a typical "distributed network" configuration (with subnets connected in a mesh pattern) grows relatively slowly with the size of the network. The configuration distribution overhead is small (under 2 Kbytes per link) since the information is essentially sent on a spanning tree and the DCDP and DRCP headers and configuration information have been carefully optimized. The periodic overhead (primarily due to YAP and DRCP periodic advertisements) grows more rapidly, but can be contained to reasonable levels by the introduction of hierarchical structures (i.e. domains) where the corresponding domains are limited to have under 100 member nodes and refreshing network metrics at most every 30 seconds.

Chapter 3: Dynamic Domain Generation: A Centralized Approach

3.1 Introduction

In recent years there has been an increasing interest in ad hoc networks. The infrastructureless nature of these networks makes them extremely important in scenarios were the fixed infrastructure cannot be used (emergency scenarios) or there is not time availability to set up the appropriate infrastructure (on-the-move military scenarios). The ability to deploy these networks quickly and have them work through rapid changes makes them ideal for battlefield and emergency situations. There have been many good solutions proposed to deal with topology management, autoconfiguration, routing and QoS in ad hoc networks; however, most of these solutions do not scale well (e.g., only to about 50 nodes). To build ad hoc networks with hundreds or even thousands of nodes, such as that required for the Future Combat System (FCS), the network must be split into relatively independent layer 3 clusters or domains.

The assumption related to the creation of layer 3 clusters or domains is that it is done after layer 2 topology management has set local parameters such as the link frequencies, spreading code, transmit power and antenna direction. At this point, when the ad hoc network is simply an interconnected mesh of potentially thousands of nodes, the domain generation divides node interfaces into different layer 3 domains. The domain generation then continuously adjusts the domain as nodes and links change to maintain good network performance.

Smaller domains allow routing, QoS and other networking protocols to operate on fewer nodes, with cross-domain interaction only through a few border nodes. This division has two key benefits. First, it reduces overall protocol overhead. In most routing protocols, for example, the route update overhead grows as $O\left(n^2\right)$ as the number of routers in a domain increases. Using smaller domains, with inter-domain interaction through a single border router per domain, we can reduce overall overhead to $O\left(n\log n\right)$. Second, if the domains are well designed, then networking protocols can be tuned to more homogenous conditions. For example, if part of the network has links constantly going up and down, then it can be put in a separate routing domain whose border router does not propagate internal changes.

The domain generation can be done using either local or global information. The two approaches are complementary since local domain generation reacts faster, requires less overhead, and is more robust; while global domain generation provides better overall domains. Most existing work on domain generation, however, has used only very limited local information. Indeed, the majority of the approaches simply elect a “cluster-head” within each subnet based on node attributes like the node ID (lowest ID [17][20], highest ID [19]) or node degree (highest degree [18]). Some proposals use local metrics during cluster generation, but the metrics are utilized just for the selection of cluster-heads; the generation of clusters is completed by assigning the non cluster-head nodes to the cluster-heads. The assignment is performed with respect to the distance (hops) between the non cluster-head nodes and the cluster-heads and the weight assigned to the latter ones [21][22].

The central objective of this work is to take into account the network environment and its dynamics, and by optimally grouping together nodes based on the appropriate metrics to improve a priori selected aspects of its performance. The proposed approach is based on a modified version of Simulated Annealing, which is a global optimization algorithm. The selection of a global optimization algorithm as the core of the domain generation algorithm can be justified from the design objectives. Even though it is expected that the algorithm will be more demanding in terms of time and network resources compared to a distributed clustering algorithm, its benefits will be observed as improvements on the selected aspects of network performance. Apart from that, the selected algorithm is very general in optimizing with respect to various objectives, which is not the case for the tailored to specific clustering objectives, distributed algorithms. The generality of the proposed clustering framework becomes more significant in cases where the network conditions change, so the clustering objectives have to change dynamically to reflect the new conditions and requirements. Due to the importance of designing a clustering algorithm able to handle such dynamic changes on the hierarchy generation objectives, the proposed algorithmic framework has been designed with respect to this specification.

The selection of general approximation algorithm (Simulated Annealing) as the core of the hierarchy generation framework inherits to the proposed approach the capability for dynamic adaptation to the performance requirements of the network.

The adaptation is performed by the application of the appropriate cost function that represents the network performance requirements at any time instance. SA operates independently of the characteristics of the cost function, which makes it a powerful algorithm. The only drawback of SA considering the environment under consideration is its slow convergence times. As it will be exploited in chapter 5, SA has been modified and adjusted appropriately so that its convergence time becomes must faster without significant loss in the quality of the provided clustering solutions. This is done by adjusting some of the core parameters of SA and modifying some of its functional components.

This work is motivated from real world examples that set the specifications for the efficient design of hierarchy generation algorithms. Most of the existing clustering techniques generate clusters without taking into consideration the network environment (i.e. link and node characteristics) since they do not aim on the improvement of specific performance aspects of the network. The justification of those techniques lies on the generation of hierarchical environment that will potentially improve the scalability or robustness of the network, since existing ad hoc protocols, like the ad hoc routing protocols, perform better when they are applied to small number of nodes, because they can capture better the dynamics of the smaller subgroups. Apart from the latter justification of why to cluster (e.g., create hierarchy), they do not impose any other stronger argument.

The proposed work has been motivated, not only from the potential advantages resulting from the application of hierarchical structures, but mainly the motivation arises from the side effects that can negatively affect the performance of the network due to the application of such structures. If the hierarchical structure is not generated appropriately with respect to the network environment and if it is not specifically customized to the network performance characteristics to be improved, then the resulted hierarchy instead of improving the functionality of the network may result in degrading its performance. The network performance could become worse compared to the corresponding performance of the flat structured network, since the maintenance of the constructed hierarchy requires extra overhead. This work targets on the minimization of this overhead through the incorporation of the network environment characteristics and the network performance aspects required to be improved, into the generation process. The objective of the approach can be described better by elaborating more on:

- the algorithm that will be responsible for the generation of the hierarchical structure, which has to be aware of the performance aspect(s) to be improved and must be able to incorporate them into the decision process
- the translation of the network environment characteristics and performance aspects of interest into entities (cost functions, metrics) that can be exploited from the introduced algorithm

The proposed approach on cluster generation differs from the existing ones on the fact that the network characteristics are taken into consideration a priori from the clustering methods. Those methods along with the appropriate metrics or combination of metrics can generate clusters that have the ability to improve the targeted performance aspects of the network. Thus, in order to take advantage of the benefits provided by the application of hierarchical structure, the network environment and the performance to be improved has to be taken into consideration in the generation process. The importance of the approach is justified from the following motivation examples.

Motivation Example I (Metric of Interest: mobility)

Assume the following network environment, where the nodes 1-7 are static sensor nodes. The nodes 8-11 are mobile nodes, which are moving as a group so they are relatively static (i.e., group of soldiers). The latter group of nodes is moving on a predefined cyclic trajectory around the sensor field.

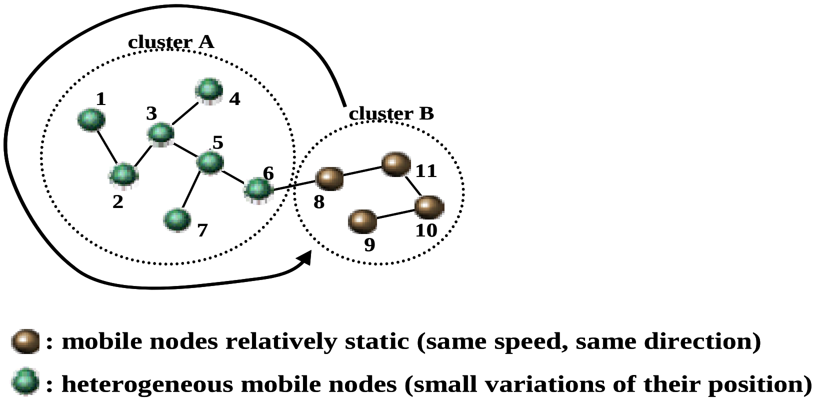

cluster A
cluster B
1
2
3
4
5
6
7
8
9
10
11
: mobile nodes relatively static (same speed, same direction)
: heterogeneous mobile nodes (small variations of their position)

Figure 3.1. Dynamic Clustering Motivation Example I (mobility vs. proximity)

In that case if we attempt to cluster based on proximity or utilizing the traditional methods of clustering (lower ID, highest degree) then we may end up with continuous re-clustering, due to the network dynamics for maintaining the clusters consistent to the chosen scheme. If instead we cluster based on the mobility of the nodes then we will not need any re-clustering because although the positions of the nodes change, their mobility characteristics remain the same.

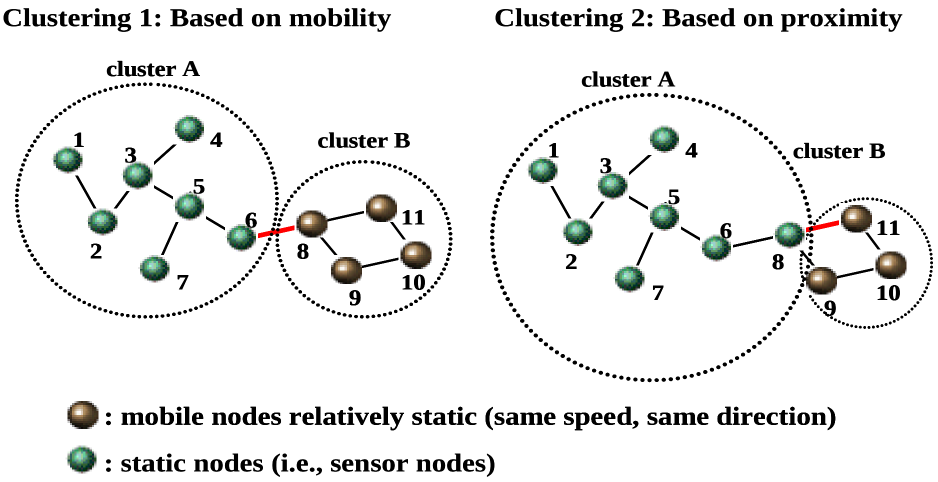

Clustering 1: Based on mobility
Clustering 2: Based on proximity
cluster A
cluster B
1
2
3
4
5
6
7
8
9
10
11
cluster A
cluster B
1
2
3
4
5
6
7
8
9
10
11
: mobile nodes relatively static (same speed, same direction)
: static nodes (i.e., sensor nodes)

Figure 3.2. Mobility vs. Proximity Based Hierarchical Structures

By taking into consideration the network environment and utilizing the mobility as criterion for cluster generation we can achieve the following:

- Minimize re-clustering/maintenance overhead
- We end up with more stable and robust to network dynamics clusters

Motivation Example II (Metric of interest: power)

In this example the network environment consists of two classes of nodes (figure 3.3). The one class involves static sensor nodes, which have been placed to strategic places on a surface for the collection of important information. These nodes are characterized from finite power. The second class involves the mobile nodes that can move towards any point on the surface. These nodes have rechargeable source of power, so they can be characterized as infinite power nodes (i.e. UAVs).

The objective in this case is to move the mobile nodes to points on the surface, where the generated clusters will improve the network performance of interest. For example in this scenario we are interested in improving the cost associated to the proximity of the nodes. The proximity of the nodes can be related to the transmission power of the nodes and the intra-cluster delay. Also, we are interested in extending the lifetime of the network. By following each one of the above directions, it is possible to end up with different clustering maps, as it is demonstrated in figure 3.3.

In the case where the cost function targets the decrease of the distance between the nodes, we may end up with a clustering map that does not protect the finite power of the sensor nodes, thus this can result in a very short network lifetime. The latter can happen because some of the border routers are finite power sensor nodes, which means that they will have to forward all the traffic for inter-cluster communication, so their available power will be drained very fast, causing network partitioning.

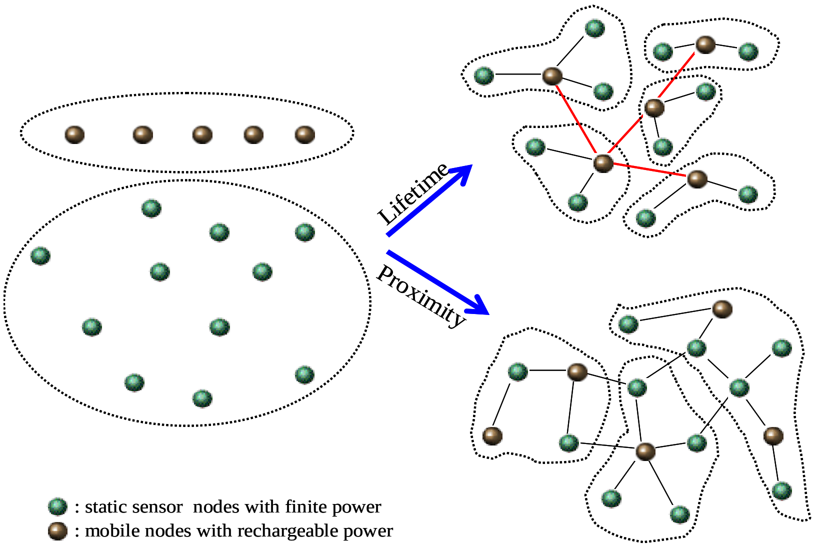

Lifetime
Proximity
● : static sensor nodes with finite power
● : mobile nodes with rechargeable power

Figure 3.3. Dynamic Clustering Motivation Example II (proximity vs. power)

By assigning the infinite power nodes in strategic places, we can extend the lifetime of the network. A promising approach would be to assign the infinite power nodes to positions where they can have connectivity with each other, and the network is connected. In other words, we create a backbone (e.g. connected dominating set) which is consisted of the infinite power nodes. Each finite power node is connected to at least one infinite power node, so the forwarding of traffic is done mostly through the backbone. The rest of the nodes (e.g., finite power nodes) save their energy, which results in prolonging the lifespan of the network.

By identifying the network environment, it is preferable to generate a hierarchy that fits better this environment and its dynamics. For the specific example, the lifespan (survivability) of the network, it is more important than the improvement of intra-cluster delay or the minimization of the number of collisions in the MAC layer due to the density of the network. The latter two aspects can be improved by grouping together the nodes that are topologically closer.

In the former example (e.g., Example I) the metric of interest is the stability of the generated clusters with respect to the mobility characteristics of the nodes. For the latter example, the metric of interest is the lifetime of the network with respect to the power constraints of the participating nodes. Obviously, depending on the environment and the requirements imposed from the functionality of the network, the hierarchy generation mechanism will adapt accordingly the network metrics being utilized depending on the corresponding cost functions. The generality of the approach proposed in this work is capable of dealing with a huge diversity of cost functions and metrics without changing its core functionality. Hence, the same algorithmic framework can be applied to generate clusters that satisfy a large variety of hierarchy generation objectives.

The importance of incorporating the network environment and the desired performance characteristics into the hierarchy generation process has been highlighted but this is not the only aspect of the proposed approach that distinguishes it from the exiting ones. There are algorithms that attempt to incorporate indirectly some of the network environment characteristics by assigning corresponding weights to the participating nodes. The assignment of weights to the nodes is done solely for the selection of clusterheads. These algorithms even though they claim their network environment awareness, they differ in two important points from the approach proposed in this work:

- The utilization of characteristics of network environment is done for the assignment of weights to the nodes so that clusterhead nodes are selected. The set of characteristics taken into consideration and their utilization do not focus on the improvement of any specific network aspect.
- The set of characteristics that are taken into consideration for the assignment of weights to the nodes cannot change dynamically. These algorithms are tailored to specific characteristics.

The algorithm proposed in this dissertation improves the existing approaches by avoiding the above weaknesses. The algorithm is not tailored to specific network characteristics (they can be selected dynamically) and operates so that the network performance due to the constructed hierarchy is improved with respect to a set of pre-specified objectives. Furthermore, the algorithm can be utilized for both the generation of multiple hierarchies at the same time that target different performance aspects of the network and a unique hierarchy that attempts to satisfy multiple design objectives. The tuning of the algorithm to the network conditions and the hierarchy generation objectives is accomplished via the appropriate adaptation of its corresponding modules (metrics, cost functions, constraints).

As we have already mentioned, the general applicability and versatility of the proposed approach results from the application of SA as the core of the hierarchy generation framework. Since SA can operate on any cost function and under any constraint, the characteristics of the generated clustering maps depend on the applied cost functions and constraints. The set of cost functions and constraints are defined with respect to the clustering objectives and the given network environment. Briefly the modules that constitute the proposed hierarchy generation framework are:

- Optimization Algorithm (Simulated Annealing)
- Metrics
- Cost functions
- Constraints

This chapter describes in detail the general approximation algorithm, namely the Simulated Annealing (SA) algorithm, which has been utilized as the cornerstone of the proposed hierarchy generation framework. Its functionality and convergence characteristics are being presented throughout the chapter. This chapter is preparatory for the two following chapters where the indicative network metrics and cost functions introduced are being presented and the modifications and adjustments applied to SA for improving its convergence characteristics are being exploited, respectively.

Section 2 overviews some of the most interesting work in the area of hierarchy generation from the mobile networks perspective. In section 3, SA algorithm is described in details (parameters and functionality) and in section 4 the constraints imposed on the hierarchical structure generation are described.

3.2 Background

The idea of generating hierarchy for scaling and making more robust the network exist for many years. In wireless ad hoc networks the idea of clustering emerged when the packet radio networks were introduced, which are the ancestors of ad hoc networks. In this area, Ephremides [19][20] introduced the idea of clustering through the concept of a distributed linked cluster architecture. The clustering objective of this work was the hierarchical application of routing in a more robust to topological changes environment. Also, the latter work was trying to take advantage of the spatial reuse and communication isolation due to clustering, by assigning different codes and frequencies to the various clusters.

The idea of clustering in ad hoc networks was revisited in the context of mobile multimedia wireless networks [17] [18]. One of the most popular clustering schemes among the existing works in the area of ad hoc networks is the Lowest-ID scheme. This scheme used in [20] as well as in [17][18]is the point of reference and of comparison for many recently introduced clustering schemes. In [17] Gerla et al., propose a simple distributed algorithm that yields clusters that are at most two hops in diameter. In each cluster the node with the lowest ID among its one hop neighbors becomes the clusterhead and maintains the cluster memberships of the other nodes in the cluster.

An algorithm based on the degree (e.g. number of 1-hop neighbors) of the nodes was proposed in [17]. The nodes having the highest degree among their 1-hop neighbors were selected to be the clusterheads. This algorithm, namely the Highest Degree clustering algorithm performed much worse than the Lowest-ID (LID) in terms of the robustness of the generated clusters. The robustness was measured by the average number of membership changes per unit of time.

After the initial approaches and the fact that the researchers were convinced about the impact and importance of hierarchy on the performance of the mobile ad hoc networks specifically, many clustering schemes have been proposed. The most significant work among other published in the area can be categorized into six classes.

The first class consists of the Dominating Set (DS) based clustering. In this class belong the Connected Dominating Set (CDS) by Wu [23] and the Weakly Connected Dominating Set (WCDS) by Chen [24] algorithms. Wu proposed a distributed algorithm for the construction of CDS in order to design efficient routing schemes for a MANET. The main objective of this scheme is to find a minimum number of nodes as dominating nodes to construct a CDS. By minimizing the number of CDS nodes, the number of nodes participating in routing decreases. The reduction of DS size is performed such that the unnecessary dominating nodes are being eliminated without breaking the direct connection between neighboring dominating nodes. Chen’s WCDS has the same objectives but it relaxes the direct connection requirement between the dominating nodes. The problem with the DS based schemes is that the network dynamics (node mobility, node failures) will cause ripple effects, so the entire topology will have to be readjusted in order to maintain the structure. Hence, such schemes are more feasible in static or low mobility networks.

The second class consists of the low maintenance clustering schemes. In this class 3 approaches are highlighted. The Least Cluster Change (LCC) [25] approach is considered to be a significant enhancement of Lowest ID Clustering (LID) or Highest Degree Clustering. In LCC, hierarchy generation is separated into two phases, the hierarchy generation which follows the LID algorithm and the hierarchy maintenance, which is event driven (i.e. when two CHs obtain a direct connection or a node cannot access any CH). Previously the hierarchy construction was executing periodically, introducing unnecessary overhead into the network. In the low maintenance clustering schemes belongs also the 3-hops Between Adjacent Clusterhead (3hBAC) [26] approach. 3hBAC forms a 1-hop non-overlapping cluster structure with three hops between neighboring clusterheads by the introduction of a new node status, named *clusterguest*. *Clusterguest* is mobile node that cannot directly connect to any clusterhead, but can access some cluster with the help of a *clustermember*. By introducing the concept of *clusterguest*, a non-overlapping cluster structure can be achieved, which can reduce the number of clusters and eliminate the ripple effect and the small unnecessary clusters formed in maintenance phase of LCC. Another low maintenance clustering scheme is the Passive Clustering (PC) [27]. PC's differs from the conventional clustering schemes, which require all the mobile nodes to advertise cluster-dependent information repeatedly to build and maintain the cluster structure, and thus clustering is one of the main sources of control overhead. PC is a clustering protocol that does not use dedicated clustering-protocol-specific control packets or signals. The elimination of explicit control messages for clustering is achieved by forming and maintaining a cluster structure only when mobile nodes have packets to send. Furthermore, PC attempts to reduce the number of gateways in order to achieve flooding efficiency.

The third class of clustering schemes consists of the mobility aware protocols. In this class belongs the MOBIC [28] clustering scheme. MOBIC suggests clusterhead election, should take mobility into consideration. For this purpose, an aggregate local mobility metric is proposed for the cluster formation process such that mobile nodes with low speed relative to their neighbors have the chance to become clusterheads. MOBIC appears to be effective for MANETs with group mobility behavior, in which a group of mobile nodes moves with similar speed and direction. However, if mobile nodes move randomly and change their speeds from time to time, MOBIC will degrade the performance of the network due to the reclustering and reaffiliation overhead.

The energy efficient clustering approaches designate the fourth class of clustering schemes for mobile ad hoc networks. Three representative approaches are being mentioned here. The ID Load Balanced Clustering (IDLBC) [30] assumes that the clusterhead nodes deplete more energy compared to the rest of the nodes, so its objective is to avoid possible node failure due to energy depletion caused by excessively shouldering the clusterhead role. The approach is based on limiting the time that a node can serve continuously as clusterhead. The major weaknesses of the approach are that the assumption may not be valid or in the case it is valid, the time alone cannot guarantee balanced depletion of the nodes' energy. Another approach of this class of schemes has been proposed by Wu [31] and is an extension of his previously Connected Dominating Set (CDS) [23] approach mentioned in the first class of schemes. The distributed construction of the CDS in [31] is energy aware by attempting to eliminate from the DS unnecessary nodes with low residual energy. The weakness of this approach is its assumption that the energy of the nodes in the DS will be depleted faster compared to the rest of the nodes. The third approach in this class of schemes has been proposed by Ryu [32]. The specific objectives of Ryu’s approach are to minimize the transmission energy consumption summed by all master-slave pairs and to serve as many slaves as possible in order to operate the network with longer lifetime and better performance. However, master node election is not adaptive, and the method of selecting the master node is not specified. Peer-to-peer communication between slaves is forbidden. In addition, the method of maintaining the cluster structure when master or slave nodes move is not addressed. Because of these restrictions, Ryu’s scheme may not be feasible for a typical MANET.

Another class of clustering schemes for MANETs is defined by the load balancing approaches. The motivation of this class’s algorithms is that there are an optimum number of mobile nodes that a cluster can handle, especially in a clusterhead-based MANET. Load-balancing clustering schemes set upper and lower limits on the number of mobile nodes that a cluster can deal with. When a cluster size exceeds its predefined limit, re-clustering procedures (i.e. merging, splitting of clusters) are invoked to adjust the number of mobile nodes in that cluster. Such an approach is followed by the Adaptive Multi-hop Clustering (AMC) [33] algorithm. AMC does not describe how the clusters are initially constructed. However, for cluster maintenance each mobile node periodically broadcasts its information, including its ID, cluster ID, and status (clusterhead/member/gateway) to others within the same cluster. By such message exchange, each mobile node obtains the topology information of its cluster. Each gateway also periodically exchanges information with neighboring gateways in different clusters and reports to its clusterhead. Thus, a clusterhead can recognize the number of mobile nodes of each neighboring cluster. AMC sets upper and lower bounds (U and L) on the number of clustermembers that a clusterhead can handle. The establishment of the values U and L is not provided in the description of the AMC algorithm. Another algorithm which is based on the load balancing principle (e.g. balance the traffic load in each cluster by limiting the number of mobile nodes that a cluster can handle around a predefined value) is the Degree Load Balancing Clustering (DLBC) presented in [30]. DLBC periodically attempts to maintain the number of mobile nodes in each cluster around a system parameter, *ED*, which indicates the optimum number of mobile nodes that a clusterhead can handle. A clusterhead degrades to an ordinary member node if the difference between *ED* and the number of mobile nodes that it currently serves exceeds some threshold value. However, since the clusterhead change is still based on node degree, DLBC likely will cause frequent re-clustering and ripple effects because of the network dynamics.

Finally, the sixth class of MANETs clustering schemes is based on the concept of combined metrics. The algorithms of this class take into account a variety if metrics, which correspond to node degree, residual energy capacity, moving speed, etc. The algorithms of this category aim on electing the most suitable clusterheads in a local area, and do not give preference to mobile nodes with certain attributes, such as other algorithms (i.e. LID, highest degree). A representative combined metrics based algorithm is the On Demand Weighted Clustering Algorithm (WCA) [21], which involves four parameters for each mobile node i in the clusterhead election procedure. These parameters are the degree-difference D_i, the sum of the distance with all neighbors P_i, the average moving speed M_i, and the clusterhead serving time T_i. The combined weight factor I_i is calculated as:

$$I_i = c_1 D_i + c_2 P_i + c_3 M_i + c_4 T_i \tag{3.1}$$

where c_1, c_2, c_3 and c_4 are the weighting factors that satisfy equation (3.2).

$$\sum_{k=1}^{4} c_k = 1 \tag{3.2}$$

In On-Demand WCA the clusterhead in a local area is chosen to be the node with the minimum combined weight factor I_i. All mobile nodes covered by elected clusterheads cannot participate in further clusterhead selection. This procedure is repeated until each mobile node is assigned to a cluster. The weakness of the WCA algorithm is its maintenance phase, which does not take into consideration the combined weight factor. This may destroy the effectiveness of the hierarchical structure.

In general, the core objective of all the above algorithms is the selection of clusterheads for the generation of clusters. The majority of them do not take into consideration the network environment for reducing the membership changes and the related overhead. Those algorithms (e.g. WCA) that they take into consideration the network environment utilize this information just for the selection of clusterheads, so they diversify the nodes. The metric utilized for the cluster formation is the proximity of the rest of the nodes (non-clusterhead) from the selected clusterhead nodes. The design objectives of the algorithms that will be introduced in this dissertation takes into consideration the network environment characteristics for the formation of robust and efficient groups. The utilization of the network environment information is not utilized to diversify the nodes (clusterhead and non-clusterhead nodes) but to group nodes that present similarities on their characteristics and objectives for establishing hierarchical structures more homogeneous and robust to network dynamics. Furthermore, the generated hierarchy does not depend on the selection of clusterhead and non-clusterhead nodes, since all the nodes are given similar importance in the process.

The only work that differs in spirit from the approaches referred above since the network environment is taken into consideration and all the nodes are given similar importance is the approach presented in [29]. This approach can be classified in the mobility aware clustering schemes and is know as (α, t) clustering. In this algorithm, a bound to the probability α of path availability is attempted to be obtained. Specifically, a mobility model was developed and used to derive analytical expressions for the probability of the path availability α with respect to time t. The proposed mobility model is a Random Walk based model which assumes that if the distance between two nodes is less than a system dependent threshold, then it is also possible to determine the conditional probability that the nodes will be within range of each other at time $t_0 + t$, given that they are within range at time t_0. Even though this is the first attempt of clustering nodes based on the characteristics of the network instead of just selecting clusterheads and generating 2-hop clusters, this probability based model fails to capture the real mobility model of nodes with respect to its neighbors and therefore it is not a sufficient approach for clustering in an environment that cannot be described from the assumed probabilistic mobility model.

The proposed clustering framework in this dissertation takes into consideration the network environment such as the mobility characteristics (speed, direction) of the nodes but the clustering decisions do not depend on metrics related to any specific mobility model. These algorithms utilize the mobility characteristics of the nodes, which are collected and processed dynamically, for robust hierarchy generation decisions. More details for the algorithmic framework that has been developed are provided in the following section.

3.3 Algorithmic Framework for Hierarchy Generation

The main objective of this work is to propose an algorithmic framework for hierarchy generation capable of capturing the characteristics of the underlying network environment and improving its performance. Adding to the challenge of designing such an algorithm, is that the algorithmic framework must be flexible and general so that it can be applied as it is in conditions of varying hierarchy generation objectives. The hierarchy generation objectives represent the network environment and the performance aspects of the network to be improved.

Additionally, due to the dynamics of the network environments under consideration (e.g. MANETs) the algorithmic framework must be capable to produce rapidly, efficient hierarchy generation solutions. The requirements on the speed of the algorithm are imposed from the degree of dynamics of the network (e.g. rate of topology changes due to mobility or node failure). In conclusion, the required characteristics of the hierarchy generation algorithmic framework to be developed are:

- Incorporate the network environment characteristics
- Generate the hierarchy for improving a set of pre-specified performance aspects
- General, so that any type of network environment and set of performance aspects can be incorporated to the algorithm without change in its functionality
- Generate efficient hierarchy solutions fast enough, so that it can capture the dynamics of networks like MANETs.

3.3.1 Combinatorial Optimization

The problem of hierarchy generation can be formulated as a combinatorial optimization problem.

Definition 3.1 A combinatorial optimization problem is either a minimization or maximization problem and is specified from a set of problem instances.

Definition 3.2 An instance of combinatorial optimization problem can be formalized as a pair (S, f), where the *solution space S* denotes the finite set of all possible solutions and the co*st function f* is the mapping defined as

$$f : S \to \mathbb{R}. \tag{3.3}$$

In the case of minimization, the problem is to find a solution $i_{opt} \in S$ which satisfies

$$f\left(i_{opt}\right) \leq f\left(i\right), \forall i \in S. \tag{3.4}$$

In the case of maximization, i_{opt} satisfies

$$f\left(i_{opt}\right) \geq f\left(i\right), \forall i \in S. \tag{3.5}$$

Such a solution i_{opt} is called a *globally-optimal* solution, either *minimal* or *maximal*, or simply an *optimum*, either a *minimum* or *a maximum*; $f_{opt} = f\left(i_{opt}\right)$ denotes the optimal cost, and S_{opt} is the set of optimal solutions.

A combinatorial optimization problem is solved by finding the “best” or “optimal” solution among a finite or countably infinite number of alternative solutions [34]. Considerable effort has been devoted to constructing and investigating methods for solving to optimality or proximity combinatorial optimization problems. Integer, linear and non-linear programming, as well as dynamic programming have seen major breakthroughs in recent years.

An important achievement in the field of combinatorial optimization, obtained in the late 1960’s, is the conjecture – which is still unverified – that there exists a class of combinatorial optimization problems of such inherent complexity that any algorithm, solving each instance of such a problem to optimality, requires a computational effort that grows superpolynomially with the size of the problem. This conjecture resulted in a distinction between hard and easy problems. During 1970’s advances in theoretical computer science have provided a rigorous formulation of this conjecture. The resulting theory of *NP-completeness* has greatly increased the insight in the relationship between hard problems.

Over the years many practical and theoretical combinatorial optimization problems has been shown that belong in the class of *NP-complete* problems. One such a problem is the graph partitioning problem (e.g. hierarchy generation). The complexity and the difficulty of the problem are increased further due to the design requirements imposed on the general applicability and the speed of the algorithm.

3.3.1.1 General Classes of Algorithms

Even though the hierarchy generation problem is *NP-complete*, still must be solved and in constructing appropriate algorithms one might choose between two options. Either one goes for optimality at the risk of very large, possibly impractical amounts of computational times or one goes for quickly obtainable solutions at the risk of sub-optimality.

The first option requires the utilization of optimization algorithms like the enumeration methods using cutting plane, branch and bound or dynamic programming techniques [34]. The second option involves approximation algorithms (heuristic algorithms). The approximation algorithms can be categorized into two classes, the local search and randomization algorithms. There are algorithms, which based on their configuration can behave as optimization or as approximation algorithms. So, between these two categories there is an overlap area. For example, by introducing heuristic bounding rules, the branch-and-bound algorithm can behave as an approximation algorithm instead of optimization algorithm.

The optimization and approximation algorithms can be further categorized into two larger classes, the tailored and general classes of algorithms. The general class involves algorithms that are problem independent, so they can solve a wide variety of problems, as opposed to the tailored class of algorithms, which are being designed to solve a specific class of problems. The intrinsic problem of tailored algorithms is that each type of combinatorial optimization problem a new algorithm must be constructed that is customized to that problem.

With respect to the design requirements imposed to the hierarchy generation algorithmic framework and by exploring the various categories of existing optimization algorithms, the best matching category is the general approximation class of algorithms. Specifically, the randomization algorithms of the latter class seem capable to provide the cornerstone for the design of the targeted hierarchy generation algorithmic framework. That is due to the general applicability of these algorithms, their speed compared to the optimization class of algorithms and the quality of the provided solutions compared to the local search class of algorithms.

The general applicability of the algorithms is important because the same framework can be applied for different network environments and hierarchy generation objectives (e.g. for the improvement of specific network performance metrics). The optimal for the same network over time might change, either due to the varying network topology or due to changes on the utilization of the network (e.g. different network objectives). In such cases we do not want to apply different hierarchy generation protocol, but it is preferable to dynamically adapt the existing one.

The speed of convergence of the approximation algorithms with respect to the optimization algorithms makes this class of algorithms more favorable for scenarios where the solution has to be obtained fast. Characteristic examples of such scenarios are the network environments under consideration (MANETs), due to their inherent dynamics (topology changes due to mobility and node failures).

The selection of randomization algorithms instead of the class of local search algorithm is justified from the quality of solutions provided from each class of algorithms. The randomization algorithms provide better solutions compared to the local search ones because the quality of the solution does not depend on the initial solution and instead of accepting only moves that result in cost improvement, at limited extend the accept also moves that result in cost deterioration. The latter is extremely important when there are local optimal points, because the randomization algorithms are able to avoid these points and converge to the global optimal solution as opposed to the local search algorithms. This is one of the most important reasons for the high quality solutions provided from the randomization algorithms.

Having specified what is the type of algorithm that best matches the characteristics of the hierarchy generation framework the next step is to locate a specific algorithm that will serve as the backbone for the design of the appropriate algorithmic framework. For this purpose, the more promising general approximation randomization algorithm is the Simulated Annealing (SA) algorithm. A modified version of SA has been used as the core for the hierarchy generation algorithmic framework, introduced in this dissertation.

3.3.2 Simulated Annealing (SA) algorithm

The (SA) algorithm was independently introduced by Kirkpatrick, Gelatt and Vecchi [35] and Cerny [36]. Other names that have been used to denote the Simulated Annealing algorithm are:

- *Monte Carlo annealing*
- *Probabilistic hill climbing*

- *Statistical cooling*
- *Stochastic relaxation*

The name “simulated annealing” originates from the analogy with the physical annealing process of solids. In condensed matter physics, annealing is known as a thermal process for obtaining low energy states of a solid in a heat bath. The process contains the following two steps:

Step 1: Increase the temperature of the heat bath to a maximum value at which the solid melts.

Step 2: Decrease carefully the temperature of the heat bath until the particles arrange themselves in the ground state of the solid.

In step 1, where the solid is in a liquid phase, all the particles of the solid arrange themselves randomly. In step 2 when the configuration of the particles reaches the ground state, the particles are arranged into a highly structured lattice and the energy of the system is minimal. The minimal energy state (ground state) of the system is obtained only if the maximum temperature is sufficiently high and the cooling is done sufficiently slow.

In 1953, Metropolis et al. [37] introduced a simple algorithm for simulating the evolution of a solid in a heat bath to thermal equilibrium. The algorithm introduced generates a sequence of states that resemble the different phases of the physical system (e.g. solid). Given a current state *i* of the solid with energy E_i, then a subsequent state *j* is generated by applying a perturbation mechanism which transforms the current state *i* into a new state *j* with energy E_j. Based on the energy difference $E_j - E_i$, the new state j is accepted of rejected probabilistically according to the following rule:

$$P\left(X_{t+1} = j \mid X_t = i\right) = \begin{cases} 1 & \text{,if } E_j < E_i \\ e^{\left(\frac{E_i - E_j}{k_B T}\right)} & \text{,if } E_j \geq E_i \end{cases} \tag{3.6}$$

where X_t, X_{t+1} are stochastic variables denoting the current and next state of the solid respectively. *T* denotes the temperature of the heat bath and k_B is a physical constant known as the Boltzmann constant. The acceptance rule (3.6) is known as the Metropolis criterion and the algorithm based on this criterion is known as the Metropolis algorithm.

In Metropolis algorithm if the temperature is being lowered sufficiently slow, the solid can reach thermal equilibrium at each temperature. Algorithmically this is achieved by generating a large number of states at each temperature. Thermal equilibrium is characterized from the Boltzmann distribution and provides the probability of the solid being in state *i* with energy E_i at temperature *T*. The Boltzmann distribution is given by

$$P_T\left\{X_t = i\right\} = \frac{1}{Z\left(T\right)} e^{\left(\frac{-E_i}{k_B T}\right)} \tag{3.7}$$

$Z\left(T\right)$ is the partition function given by

$$Z\left(T\right) = \sum_j e^{\left(\frac{-E_j}{k_B T}\right)} \tag{3.8}$$

where *j* represents all the possible states of the solid.

The Metropolis algorithm, the corresponding Metropolis criterion and Boltzmann distribution play central role in the description of Simulated Annealing algorithm, since they constitute its core functional module. The Metropolis algorithm was designed to simulate the evolution of a solid in a heat bath, but the Simulated Annealing algorithm was introduced in a more general framework for the solution of combinatorial optimization problems. The analogy of combinatorial optimization problem with a physical many particle system is defined from the following equivalences.

- Solutions in a combinatorial optimization problem are equivalent to states of a physical system.
- The cost of a solution is equivalent to the energy of a physical system's state.

Important role to the effectiveness of the physical annealing process plays the temperature (e.g. initial temperature, cooling schedule). In SA the role of temperature is resembled from the control parameter c, which is also very important for the quality of the obtained optimization solutions. At each temperature a number of solutions are generated following the transition mechanism of SA. A transition is defined as:

Definition 3.3 A transition is a combined action resulting in the transformation of a current solution into a subsequent one. The action consists of the following two steps: (i) application of the generation mechanism, (ii) application of the acceptance criterion.

The acceptance criterion is defined in analogy to the metropolis criterion in Metropolis algorithm.

Definition 3.4 Let (S, f) denote an instance of a minimization combinatorial optimization. If i is the current solution with cost $f(i)$ and j is the new solution generated from *i* with cost $f(j)$, then the acceptance criterion determines whether *j* is accepted from *i* by applying the following acceptance probability:

$$P_c(\text{accept } j \text{ from } i) = \begin{cases} 1 & \text{, if } f(j) < f(i) \\ e^{\left(\frac{f(i)-f(j)}{c}\right)} & \text{, if } f(j) \geq f(i) \end{cases}, \tag{3.9}$$

where $c \in \mathbb{R}^+$ denotes the control parameter.

In the case of maximization combinatorial optimization problem the acceptance criterion is the same by substituting the cost function values with their negative values.

The pseudo C algorithm that describes the SA algorithm is provided in figure 3.4. Some of the important parameters portrayed in this algorithm are the c_k and L_k, which denote the value of the control parameter and the number of transitions generated at the k^{th} iteration, respectively. Parameter i_{start} represents the initial solution bootstrapping the SA algorithm.

```
function SIMULATED_ANNEALING()
{
    INITIALIZE($i_{start}, c_0, L_0$);
    $k = 0$;
    $i = i_{start}$;
    do{
        for $(l = 1; l <= L_k; l++)$
        {
            GENERATE( $j$ from $S_i$ );
            if $(f(j) < f(i))$
                $i = j$;
            else
                if $\left( e^{\left(\frac{f(i)-f(j)}{c_k}\right)} > random[0,1) \right)$
                    $i = j$;
        }
        $k++$;
        CALCULATE_LENGTH$(L_k)$;
        CALCULATE_CONTROL$(c_k)$;
    }while stopcriterion
}
```

Figure 3.4. Pseudo C description of SA algorithm

A strong feature of SA, with respect to the initial solution i_{start}, is that it finds high quality solutions which do not strongly depend on the choice of i_{start}. Due to this attribute the algorithm is characterized effective and robust. This characteristic mainly results from a typical feature of SA, where the algorithm besides accepting improvements in cost, it also to a limited extend accepts deteriorations in cost based on the acceptance criterion. For large values of the control parameter *c* the probability of accepting deteriorations in cost is high. As the value of *c* decreases towards 0, the acceptance probability for deterioration in cost decreases and becomes 0 when the control parameter *c* becomes also 0. The latter feature of SA is important both for the asymptotic convergence of the algorithm to the global optimal, since the algorithm can escape from local minima. Because of this feature the algorithm combines the simplicity and generality of local search methods and the ability to provide high quality solutions, since it has the mechanism to avoid low quality local minima solutions.

The ability of SA to escape local minima using the acceptance criterion (3.6) and the Boltzmann distribution (3.7) play the most important roles for the asymptotic behavior of the algorithm. The SA algorithm, which belongs in the class of randomization general approximation algorithms, asymptotically behaves as an optimization algorithm, since it converges to the set of globally optimal solutions. Specifically, the following conjecture and corollary formalize better the asymptotic global optimality provided from the algorithm.

Conjecture 3.1 Given an instance $\left(S, f\right)$ of combinatorial optimization problem and a suitable neighborhood structure then, after sufficiently large number of transitions at a fixed value of *c*, applying the acceptance probability (3.6), the simulated annealing (SA) algorithm will find a solution $i \in S$ with a probability equal to

$$P_c\left\{X = i\right\} \overset{def}{=} q_i\left(c\right) = \frac{1}{N_0\left(c\right)} e^{\left(-\frac{f(i)}{c}\right)} \tag{3.10}$$

where X is a stochastic variable denoting the current solution obtained by the simulated annealing algorithm and

$$N_0\left(c\right)=\sum_{j\in S}e^{\left(-\frac{f(j)}{c}\right)} \tag{3.11}$$

denotes a normalization constant.

The neighborhood structure depends on the generation mechanism one is applying to obtain new solutions. More details on this follow, when the adjustment of the various parameters of the algorithm is presented. The probability (3.10) is also called the stationary or equilibrium distribution and the normalization constant $N_0\left(c\right)$ is equivalent of the partition function $Z\left(T\right)$ given from (3.8).

Before stating the corollary for the asymptotic global optimality of the solution provide from SA algorithm, we define the characteristic function $\chi_{(A')}$ of the set A'.

Definition 3.5 Let A and $A'\subset A$ be two sets. Then the characteristic function $\chi_{(A')}:A\rightarrow\{0,1\}$ of the set A' is defined as $\chi_{(A')}\left(\alpha\right)=1$ if $\alpha\in A'$, and $\chi_{(A')}\left(\alpha\right)=0$, otherwise.

Corollary 3.1 Given an instance $\left(S,f\right)$ of a combinatorial optimization problem and a suitable neighborhood structure. Furthermore, let the stationary distribution be given by (3.10), then

$$\lim_{c\rightarrow 0}q_i\left(c\right)\overset{def}{=}q_i^*=\frac{1}{\left|S_{opt}\right|}\chi_{\left(S_{opt}\right)}\left(i\right) \tag{3.12}$$

where S_{opt} denotes the set of globally optimal solutions.

Proof:

Using the fact that $\forall \alpha \le 0$, $\lim_{x \to 0} e^{\frac{\alpha}{x}} = 1$ if $\alpha = 0$, and 0 otherwise, we obtain

$$\lim_{c \to 0} q_i(c) = \lim_{c \to 0} \frac{e^{\left(-\frac{f(i)}{c}\right)}}{\sum_{j \in S} e^{\left(-\frac{f(j)}{c}\right)}}$$

$$= \lim_{c \to 0} \frac{e^{\left(\frac{f_{opt} - f(i)}{c}\right)}}{\sum_{j \in S} e^{\left(\frac{f_{opt} - f(j)}{c}\right)}}$$

$$= \lim_{c \to 0} \frac{1}{\sum_{j \in S} e^{\left(\frac{f_{opt} - f(j)}{c}\right)}} \chi_{\left(S_{opt}\right)}(i) + \lim_{c \to 0} \frac{e^{\left(\frac{f_{opt} - f(i)}{c}\right)}}{\sum_{j \in S} e^{\left(\frac{f_{opt} - f(j)}{c}\right)}} \chi_{\left(S \backslash S_{opt}\right)}(i)$$

$$= \frac{1}{\left|S_{opt}\right|} \chi_{\left(S_{opt}\right)}(i) + 0$$

■ **Q.E.D.**

Corollary 3.1 guarantees asymptotic convergence of the SA algorithm to the set of globally optimal solutions under the condition that the stationary distribution of (3.10) is attained at each value of the control parameter *c*. The result of this corollary is very important with respect to the asymptotic behavior of the SA algorithm and its capability to obtain high quality solutions.

The ability of SA to provide high quality solutions is one of the characteristics that make the algorithm favorable as the backbone of the clustering framework developed in this dissertation. The other important characteristic of SA that makes the algorithm suitable is the generality of the algorithm with respect to the cost functions optimized and the constraints imposed to these cost functions. Particularly, SA algorithm is able to deal with cost functions with quite arbitrary degrees of non-linearity, discontinuity, and randomness and can process quite arbitrary boundary conditions and constraints imposed on these cost functions. The significance of the latter characteristic of SA is located to the generality of the proposed clustering framework. The decoupling of the algorithmic details from the clustering objectives and the characteristics of network environment can be achieved by utilizing any type of cost function and constraint on demand, which represent a set of clustering objectives and the network environment dynamics. Since these parameters might change dynamically during the course of the network lifetime, the cost functions and constraints representing them will also have to change. The clustering algorithm has to be able to optimize the new cost functions under the new constraints without any modifications on the functionality of the algorithm. SA can provide this flexibility due to its generality and the high quality of the clustering solutions it can obtain.

Simulated Annealing algorithm has been applied successfully for the solution of complex optimization problems in various areas and it has been shown to perform well with respect to the quality of the solutions obtained. Some of the performance results, which are consistently reported, are:

- SA outperforms time equivalent local search algorithms (LSA) with respect to the quality of the provided solutions. Time equivalent local search algorithms run multiple times during a single run of SA, by starting from different initial solutions.

- For engineering problems (image processing, VLSI design, code design) there no tailored algorithms exist, so SA is a panacea because solves complex problems by converging to high quality solutions.
- In graph partitioning problems SA behaves better with respect to error and running time than classical edge interchange algorithms introduced by Kerningham and Lin [37].

Apart from the consistent performance results of SA, there are have been reported many that are inconsistent between different studies. The inconsistencies have to do with the convergence times and degree of suboptimality of the provided solutions on solving the same problem. SA algorithm is a simple to implement powerful tool for solving optimization problems. The simplicity of the implementation is responsible for the performance evaluation inconsistencies, since the various modules of the algorithm, which solve the same problem, could have been implemented in many different ways, resulting in different convergence times and solution qualities. In general, experience shows that SA performance depends on the skill and effort of its application. In this dissertation we focus on the latter observation, in order to implement and configure appropriately for its application in dynamic environments (e.g. fast convergence times allowing a small degree of suboptimality). More details on the configuration of SA are provided in chapter 5, after the presentation of the cost functions introduced in chapter 4. This is because part of the configuration process depends on the clustering objectives (cost functions), so we have initially to give the set of cost functions applied and then to describe the customization process of the various SA parameters.

3.4 Topological Constrains

Apart from the optimization algorithm (SA) and the cost functions, the developed clustering framework also consists of constrains enforced to the generated hierarchical structures. The main constrain imposed is the generation of topological clusters. Topological clusters are defined as:

Definition 3.6 (**Topological Cluster**): A cluster consisting of the set $\mathbb{S}$ of nodes is called topological if $\forall node_i, node_j \in \mathbb{S}$ and $i \neq j$, there is always a path P_{ij} from $node_i$ to $node_j$ such that $\forall node_k \notin \mathbb{S}$ holds that $node_k \notin P_{ij}$. All the members of a cluster can communicate between them without the need to use inter-cluster links, which are links that involve non-member nodes.

The clusters that satisfy this constrain are also called *feasible clusters*, using the optimization algorithms terminology. Example of topological (feasible) and not topological clusters (non feasible) clusters is provided in Figure 3.5.

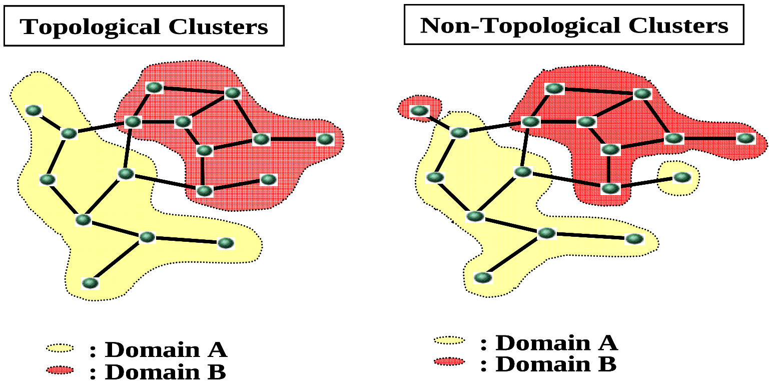


Figure 3.5. Topological (feasible) and non topological (non feasible)

The group of nodes that constitute the topological clusters is better isolated from the rest of the groups. This structural constrain is important for the aggregation and abstraction of the clusters for reducing the utilization of network resources (e.g. less communication and control information is required by the applied hierarchical protocols since each group of nodes can be represented from a single node). Also the isolation can improve the security of the network since each group can be shielded better from various attacks. The latter characteristic is an indicative example of the easier network manageability due to the existence of topological clusters.

Non topological clusters are considered the logical clusters which are constructed without taking into consideration the topology of the network. The construction of such clusters is mainly based on the logical characteristics of the nodes (i.e. the rank or the squad of the users in the case of military networks). Even though we do not consider this class of clusters in this dissertation, the same algorithmic framework could be utilized for the generation of logical hierarchical structures. The only change would be the removal of the topological clusters constraint and the introduction of the appropriate cost function that represent the corresponding logical clustering objectives and are based on the logical characteristics of the participating nodes.

Chapter 4: Dynamic Domain Generation: Metrics and Cost Functions

4.1 Introduction

Apart from the optimization algorithm, the hierarchy generation framework consists of more components:

- Collection of the Appropriate Metrics
- Definition of Cost Functions
- Constraints

The main thrust of this work is the optimization of the generated hierarchy with respect to a set of pre-specified objectives related to the performance improvement of the network. The generality provided by the utilization of SA, uncouples this work from specific metrics and objectives, as opposed to other algorithms. The hierarchy generation objectives are selected dynamically, depending on factors, like the network environment and the performance of the applied protocols. The objectives selected have to be translated into a form coherent to the optimization algorithm. Such form is the outcome of the translation of objectives into mathematical formulas (cost functions) that represent the cost of a specific hierarchical configuration. If the translation is accurate then the optimization of the cost functions will result into clustering maps that satisfy the targeted objectives. The success of the scheme is based on defining the appropriate cost functions that represent precisely the performance objectives to be accomplished. In this chapter two sets of indicative cost functions are introduced and evaluated. The one set is related to the physical structure of the generated clusters and the other set corresponds to the robustness (stability) of the constructed hierarchy with respect to mobility. There are also cost functions that represent only one clustering objective or a combination of these objectives (multi-objective cost functions). The latter group has significant importance in environments where multiple protocols are applied and the combined optimization of their performance is required.

The building blocks of the cost functions are parameters (metrics) that can be collected in real time from the network and are related to the targeted objective(s). The definition and selection of metrics of interest are important, so that by combining them with the hierarchy generation objectives, the appropriate representative cost functions can be introduced. The real time availability of the metrics is important for the application of the hierarchical generation framework in dynamic environments. In this chapter we present two sets of metrics that can be measured online from the network, related to the characteristics of the generated clusters and the mobility of the participating nodes. The cost functions emerging from these sets of metrics are presented and their ability to meet the pre-specified objectives is evaluated in the performance evaluation section.

4.2 Metrics

The SA algorithm even though is the core of the proposed scheme, is irrelevant with the performance objectives required to be met from the generated hierarchy. SA provides the optimization framework but the linkage of this framework with the network environment is done through the application of the appropriate cost functions. The optimization of these cost functions results in the hierarchical structure that satisfies the performance objectives. The building blocks of the cost functions are various metrics (parameters) that describe in a mathematical fashion the characteristics of the network environment. This section presents the most important metrics that have been applied for defining the cost functions introduced in this dissertation.

The set of metrics presented in this section can be classified into two large categories. In the first category belong the metrics that are related to the characteristics of the generated clusters. We refer to this category as cluster-information metrics. The second category involves metrics that are related to the characteristics of the participating nodes and more specifically are related to their mobility attributes. We refer to the second category of metrics as node-mobility metrics. The in-depth description of metrics is given in the following two subsections.

4.2.1 Cluster-Information Metrics

This class of metrics represents information related to the characteristics of the generated clusters. The utilization of metrics of this type in the definition of cost functions aims on the construction of hierarchical structures that consist of clusters with specific properties with respect to their topological characteristics. The cluster-information metrics introduced into the cost functions are:

- Cluster size
- Cluster diameter
- Number of border routers

More in depth explanation of the metrics and their importance in the construction of cost functions are given in the following paragraphs.

Cluster Size

The size of a cluster depends on its membership information. The number of nodes that constitute the cluster determines its size. A more precise definition of the metric is:

Definition 4.1 (Cluster Size): If the i^{th} cluster C_i consists of the nodes $n_{C_i} : n \in C_i$ then the size of the cluster C_i is given from its cardinality $|C_i|$.

Some examples related to the utilization of cluster size metric as part of hierarchy generation objectives are the construction of balanced size clusters and the generation of clusters with specific requirements on their size (e.g. lower and/or upper bounded size). The enforcement of such requirements on the generated hierarchy may target network performance objectives related to the behavior of the applied networking protocols, like the routing or MAC protocols. Specifically, it has been shown that some MANET routing protocols can perform optimally when applied to specific size networks. Taking into consideration metrics like the cluster size can be very helpful for scenarios like the latter one, where different instances of the same routing protocol can be applied in the various generated clusters. These clusters could have been sized with respect to the optimal functionality of the corresponding routing protocol.

Cluster Diameter

The cluster diameter is characterized from the longest path, with respect to the number of hops, defined between nodes that belong in the corresponding cluster. A more precise definition of the metric is:

Definition 4.2 (Cluster Diameter): If the distance in number of hops between two nodes n_k, n_w is d_{n_k, n_w}, then the i^{th} cluster's C_i diameter d_{C_i} is defined between the nodes $n_k^i, n_w^i \in C_i$ such that $d_{C_i} = \max_{n_k^i, n_w^i \in C_i} \left(d_{n_k^i, n_w^i} \right)$

The utilization of cluster diameter metric in the design of cost functions can be useful when the hierarchy design objectives are related to the diameters of the generated clusters. Sample objectives of this kind could be the design of balanced diameter clusters or clusters that have to be bounded in terms of their diameter. As in the case of cluster size metric, objectives related to the cluster characteristics may be required for improving the functionality of the various networking protocols (routing, MAC, security). For example, by imposing specific requirements on the cluster diameter could affect the overhead of the applied routing protocol (i.e. limit the flooding for the establishment of a path in the case of reactive (on-demand) routing protocols). Specific examples of the application of the metric into cost functions will be given in the next section, when we present the corresponding cost functions.

Number of Border Routers

The nodes that have links to more than one cluster are considered to be border routers. Each cluster has a set of such nodes. The size of this set defines the number of border routers for the specific cluster. A more precise definition of this metric is:

Definition 4.3 (Number of Border Routers): If $l_{n_k^i, n_w^j}$ defines a link between node $n_k^i \in C_i$ and node $n_w^j \in C_j$ and $L_{n_k^i} = \left\{ l_{n_k^i, n_w^j} \right\}$ is the set of links of node $n_k^i \in C_i$, then the set of border routers B_r^i of i^{th} cluster C_i is defined as:

$$B_r^i = \left\{ n_k^i : \exists l_{n_k^i, n_w^j} \in L_{n_k^i}, i \neq j \right\} \tag{4.1}$$

The cardinality $\left| B_r^i \right|$ of B_r^i defines the number of border of routers for the C_i cluster.

Similarly to the previously mentioned cluster information metrics, the number of border routers can be useful when the performance objectives are related to the topological characteristics of the generated clusters. Indicative example for the usage of the number of border routers metric is when the size of the set B_r^i for each cluster needs to be controlled. The requirement for specific number of border routers could be imposed from various network performance objectives. Examples of such performance objectives could be the load balancing of inter-cluster traffic (i.e. more border routers is better) or the isolation of the cluster with respect to security (i.e. small number of border routers is better). Specific usage examples of the metric as part of cost functions are provided in a later section where the designed cost functions are explained.

4.2.2 Node-Mobility Metrics

As opposed to the previous class of metrics, this class involves metrics that are related to the characteristics of the participating nodes. Specifically, the metrics are related to the mobility of the nodes and their high significance can be justified from the characteristics of the network environments under consideration. In MANETs the traditional networking protocols fail due to mobility. The newly introduced protocols cannot handle the dynamics of these networks. Thus, mobility is very important when the performance of MANETs is considered, and it is definitely one of the most significant characteristics to be taken into account for efficient hierarchy generation. This section elaborates on the node-mobility metrics, which have been introduced through this work, in the generation of hierarchical structures. Namely these metrics are:

- Node Speed
- Node Direction and Relative Direction
- Node Velocity and Relative Velocity
- Link Expiration Time

The in depth explanation of these metrics will be presented in the following paragraphs along with some representative applications of them with respect to the generation of hierarchical structures that meet a set of pre-specified objectives.

Node Speed

The speed S_{n_k} of a node n_k is a scalar quantity which refers to "how fast an object is moving." A fast-moving object has a high speed while a slow-moving object has a low speed. An object with no movement at all has a zero speed. A more precise definition of the speed of a node is:

Definition 4.4 (Node Speed): The speed S_{n_k} of a node n_k is defined as the magnitude of the distance that is covered by n_k in a unit of time (i.e. meters per second).

The speed of the node at any instance time is only part of its mobility characteristics, but it provides an indication of the impact that this node can have on the dynamics of the network. Even though, the mobility level of a node can be determined out of the speed metric, the precise impact of this node can be determined only by collecting knowledge related to the rest of the nodes into the network (e.g., even if a node presents high speed, it may by static with respect to the rest of the nodes). This metric is important for the evaluation of other more precise metrics related to the mobility of the nodes. A node can determine its speed utilizing tools like Global Positioning System (GPS) devices or tachometer sensors.

Node Direction and Relative Direction

The direction θ_{n_k} of the node n_k determines the coordination of the node's movement into the space. It can be defined as the space between two lines or planes that intersect or as the inclination of one line to another. The following figure and the precise definition of the metric provide a better understanding of direction and its significance in describing the mobility characteristics of a node.

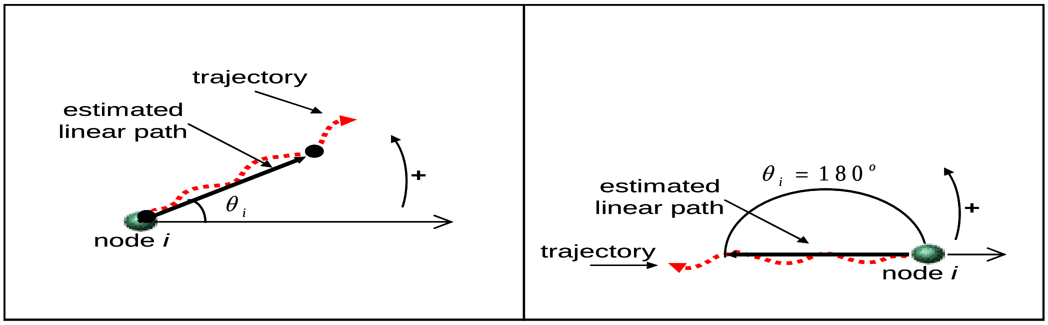


Figure 4.1. Definition and estimation of the node’s direction

Definition 4.5 (Node Direction): If the trajectory $t_r^{n_i}$ of a node n_i, belongs to a rectangular Cartesian coordinate system $(xy - plane)$, the direction θ_i of a node n_i at the point (x_1, y_1), can be described as the counter-clockwise angle represented from the straight line defined from two consecutive points (x_1, y_1), (x_2, y_2) on the trajectory of the node and the straight line $y = y_1$. The direction θ_i is measured in degrees or radians.

The direction of a node and its speed suffice to describe its instantaneous mobility characteristics. The utilization of this metric is very important whenever the design objectives are related with the mobility of the participating nodes. Even though the application of this metric alone may not be sufficient for the construction of effective cost functions representing the various clustering objectives, but similarly to the significance of the speed metric, the direction of a node is utilized indirectly in the computation of other more complete node-mobility metrics.

A more complete metric in terms of characterizing the dynamics of the network, is the relative direction $\theta^{r}_{n_i,n_j}$ between a pair n_i, n_j of the participating nodes. The completeness of this metric compared to the node's direction or speed is due to the fact it captures the relative mobility characteristics of the participating nodes. By taking into account the direction of a single node, no conclusions can be drawn about the network dynamics. Hence, the mobility characteristics of the various nodes have to be evaluated with reference to the rest of the participating nodes. The evaluation of the more complete metrics like the relative direction requires knowledge about the individual mobility characteristics of the nodes – for this specific metric knowledge of the nodes' direction is required. The relative direction provides an estimate of the relative coordination of movement between two nodes. When the value of the metric is large then the larger is the difference in the direction of the nodes, which results in higher probability that these nodes if they are connected, they will get disconnected soon or if they are not connected they will not establish a direct communication link. The opposite happens when the value of the metric is small. A more comprehensive definition of the relative direction metric is given with respect to figure 4.2.

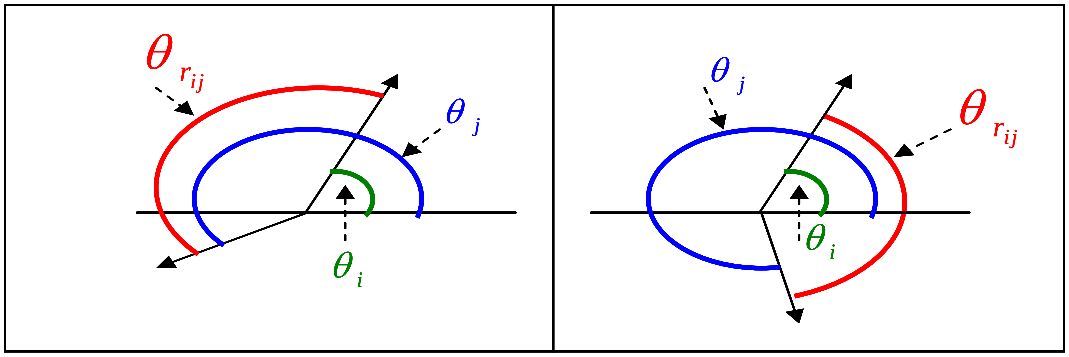


Figure 4.2. Definition and estimation of the relative direction between the nodes i, j

Definition 4.6 (Relative Direction): If a node n_i is moving with direction θ_i and a node n_j is moving with direction θ_j then we define as the relative direction $\theta_{r_{ij}}$ between the nodes n_i , n_j the angle:

$$\theta_{r_{i,j}} = \min\left(\left|\theta_i - \theta_j\right|, 360 - \left|\theta_i - \theta_j\right|\right), \tag{\textbf{4.2}}$$

where $\theta_{r_{ij}}$ is measured in degrees and $\theta_i, \theta_j \in \left[0^o, 360^o\right)$, $\theta_{r_{i,j}} \in \left[0^o, 180^o\right]$. In case where the relative direction is measured in radians the definition becomes:

$$\theta_{r_{i,j}} = \min\left(\left|\theta_i - \theta_j\right|, 2\pi - \left|\theta_i - \theta_j\right|\right), \tag{4.3}$$

where,

$\theta_{r_{ij}}$ is measured in radians and $\theta_i, \theta_j \in \left[0, 2\pi\right)$, $\theta_{r_{i,j}} \in \left[0, \pi\right]$

The value of the relative direction for two nodes partially characterizes the similarity of their movement. The smaller is the value of $\theta_{r_{ij}}$ the more similar is the movement of the nodes. This characterization is partial since the mobility of the nodes depends also on their speeds. An indicative objective related to the relative direction of the participating nodes is the grouping of nodes that present similar mobility characteristics. Such grouping can provide more robust hierarchy, which will result in fewer membership changes and less overhead related to the maintenance of the clusters. The nodes can compute their relative direction to the rest of the nodes by just exchanging their direction of movement. A node can get information about its direction by utilizing tools such as compasses or GPS devices.

Node Velocity and Relative Velocity

The relative direction metric, can partially characterize the mobility of a node with respect to others. The partial characterization is due to the lack of speed parameter in the definition of the relative direction. In this paragraph we present a more useful metric that combines the speed and direction characteristics of the nodes. Prior to presenting the relative velocity metric, the node velocity metric is defined. The velocity of a node is a vector quantity and not scalar as the speed is. The velocity, as opposed to speed, is "aware" of the direction of the node. The node velocity at any instance is characterized from the speed of the node and the vector of its direction. The formal definition of the metric is provided with respect to figure 4.3.

In the following graph d represents the distance of the node and t the time. The distance covered from a node is a function of time $d = f(t)$. For the specific time instances t_1, t_2, the distances covered from a node can be expressed as $d_1 = f(t_1)$, $d_2 = f(t_2)$ respectively. The time instance t_2 can expressed with respect to t_1 as $t_2 = t_1 + \Delta t$, where the difference $\Delta t = t_2 - t_1 \to 0$. On the graph, for smaller and smaller values of Δt, the slope of the line segment through (t_1, d_1) and

(t_2, d_2) should approach that of a tangent line touching the curve at the point $(t_1, d_1) = (t_1, f(t_1))$. With

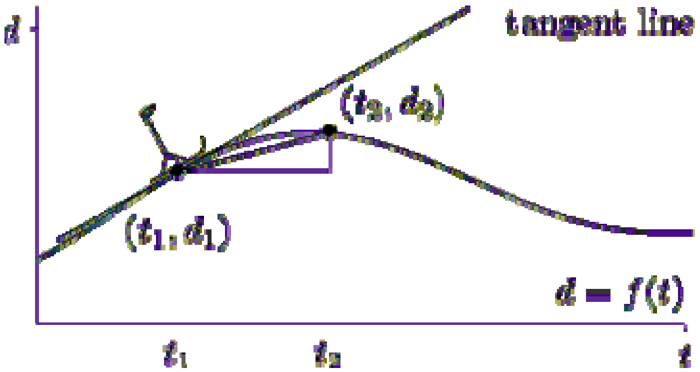


Figure 4.3. A sample distance-time graph for defining the velocity of a node

respect to the latter description we define the slope m at the point $(t_1, d_1) = (t_1, f(t_1))$ of the above curve.

Definition 4.7 (Slope of a curve at a fixed point): The slope $m = m_{\text{tangent}}$ of the tangent line through $(t_1, d_1) = (t_1, f(t_1))$ is defined by:

$$m = m_{\text{tangent}} = f'(t_1) = \lim_{\Delta t \to 0} \frac{f(t_2) - f(t_1)}{t_2 - t_1} = \lim_{\Delta t \to 0} \frac{\Delta d}{\Delta t} \tag{4.4}$$

The slope m_{tangent} defines the limiting value of average velocity over the time interval t_1 and $t_2 = t_1 + \Delta t$ as $\Delta t \to 0$. This physical interpretation provides motivation for the definition of node velocity.

Definition 4.8 (Node Velocity): The velocity u_{n_i,t_1} of a node n_i at a time instant t_1 is given by:

$$u_{n_i,t_1} = m_{\text{tangent}} = f'(t_1) = \lim_{\Delta t \to 0} \frac{f(t_2) - f(t_1)}{t_2 - t_1} = \lim_{\Delta t \to 0} \frac{\Delta d}{\Delta t} \tag{4.5}$$

Due to the speed and direction "awareness" of the metric, can be considered more complete in describe the mobility characteristics of a node compared to direction or speed. A node can determine its velocity in the same fashion it determines its speed and direction, by utilizing devices like a tachymeter, a compass or a GPS module.

On the other hand, as is the case with node speed and node direction metrics, the node velocity metric cannot capture by itself the dynamics of the network due to mobility. For this to happen, the values of the metric for the various participating nodes have to be evaluated in correlation with each other. Thus, the metric of relative velocity is introduced to eliminate the weakness of the node velocity. The relative velocity $u^r_{n_i,n_j}$ of the nodes n_i, n_j characterizes the difference of the movement patterns of these nodes. A more precise definition of the metric is:

Definition 4.9 (Relative Velocity): The relative velocity $u^r_{n_i,n_j}$ of two nodes n_i, n_j is the velocity with which the one node approaches or recedes from the other node. If the nodes n_i, n_j have speeds S_{n_i}, S_{n_j} and directions $\theta_{n_i}, \theta_{n_j}$ respectively, then the relative velocity is defined as:

$$u^{r}_{n_i,n_j} = \sqrt{\left(\Delta S^{X}_{n_i,n_j}\right)^2 + \left(\Delta S^{Y}_{n_i,n_j}\right)^2} \tag{4.6}$$

where $\Delta S^{X}_{n_i,n_j}$ and $\Delta S^{Y}_{n_i,n_j}$ are the differences of the nodes speeds on the x-axis and y-axis respectively. The $\Delta S^{X}_{n_i,n_j}$, $\Delta S^{Y}_{n_i,n_j}$ are defined as:

$$\Delta S^{X}_{n_i,n_j} = S_{n_i} \cos\theta_{n_i} - S_{n_j} \cos\theta_{n_j} \tag{4.7}$$

$$\Delta S^{Y}_{n_i,n_j} = S_{n_i} \sin\theta_{n_i} - S_{n_j} \sin\theta_{n_j} \tag{4.8}$$

By substituting (4.7) and (4.8) in (4.6) we obtain a more detailed definition of relative velocity:

$$u^{r}_{n_i,n_j} = \sqrt{(S_{n_i} \cos\theta_{n_i} - S_{n_j} \cos\theta_{n_j})^2 + (S_{n_i} \sin\theta_{n_i} - S_{n_j} \sin\theta_{n_j})^2} \tag{4.9}$$

The larger is the value of the metric, the more different is the movement of the nodes and vice, versa. The value of the metric characterizes the stability of the link between the nodes, if a communication link exists. If such a link does not exist, the value of the metric can be indicative for the stability of the network topology. The relative velocity can be utilized in cases where the clustering design objectives are related to the robustness of the generated hierarchy. Similarly, to the relative direction metric, the relative velocity could you be used for the generation of robust clusters, which consist of nodes with similar velocities (e.g. small relative velocities). Such objectives have been used for the design of some of the cost functions that will be presented in the following section. The significance of generating robust clusters, with respect to their membership, is due to the expected reduction on the hierarchy maintenance overhead. The hierarchy maintenance overhead is not related only to the managing and reforming of the hierarchical structure but also and most importantly to the effect that it has on the functionality of the applied networking protocols (i.e.

require reconfiguration – they have to go through a transient state until they reach again their steady state operation – these transitions can cause irregularities in the functionality of the entire network). The less are the number of membership changes, the better the performance and the more stable the network is expected to be.

The relative velocity can be computed with respect to the knowledge of the node velocity metric. The participating nodes can exchange their velocity (speed and direction) characteristics and then locally, using (4.9), can compute their relative velocity with any other node into the network.

Link Expiration Time

One more interesting mobility metric utilized in the construction of cost functions is the Link Expiration Time (LET) metric. Even though this metric is related to the life expectancy of a link between two nodes n_k, n_j, it has been included in the node-mobility class of metrics, because it depends on the mobility characteristics of the corresponding nodes. Namely, it depends on the speed and direction of the nodes. LET can be defined as the estimated lifetime of a link that exists between two nodes n_k, n_j, given their speeds and directions. A more precise definition of the metric is given with respect to figure 4.4.

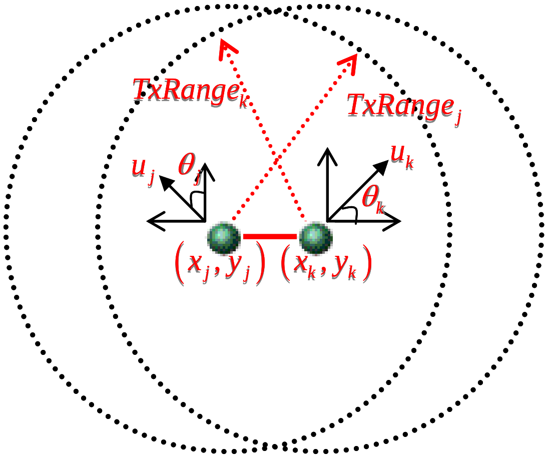


Figure 4.4. Speed, direction and transmission range characteristics between two directly communicating nodes n_k, n_j.

Definition 4.10 (Link Expiration Time): If for a time instance t, u_j, u_k characterize the speeds, θ_j, θ_k characterize the directions, $TxRange_j$, $TxRange_k$ characterize the transmission ranges and $(x_j, y_j), (x_k, y_k)$ are the coordinates of the nodes n_j, n_k respectively, the link expiration time LET_{jk} for the direct link between these nodes n_j and n_k is defined as:

$$LET_{jk} = D_{t(j \leftrightarrow k)} = \begin{cases} \dfrac{-(ab+cd)+\sqrt{(a^2+b^2)r^2-(ad-bc)^2}}{a^2+c^2} & \text{,nodes } j,k \text{ are in range} \\ 0 & \text{,nodes } j,k \text{ are not in range} \\ \infty & \text{,nodes } j,k \text{ are relatively static} \end{cases} \qquad (4.10)$$

where

$$a = u_j \cos\theta_j - u_k \cos\theta_k$$
$$b = x_j - x_k$$
$$c = u_j \sin\theta_j - u_k \sin\theta_k$$
$$d = y_j - y_k$$
$$r = TxRange_{j,k} \quad \text{(}TxRange\text{ in this case is assumed the same for every node)}$$

Similar to the relative velocity metric, LET is aware of both the speed and direction of the nodes. Thus, the metric can be utilized in cost functions that attempt to meet hierarchy generation objectives related to the dynamics of the network. Such an objective is the reduction of hierarchy maintenance overhead for the reasons mentioned above. The fulfillment of such an objective can be realized by generating robust clusters with respect to the network dynamics. In terms of LET metric, the latter objective can be translated as the grouping of nodes that define links with large LET values. The larger is the LET value for a link, the longer are expected to remain connected the nodes that define it.

The metrics presented in this section are the building blocks for the design of representative cost functions, whose optimization will generate the hierarchy that satisfies a set of pre-specified performance objectives. The set of cost functions that has been proposed in this work along with the corresponding performance objectives that have been served as the source of inspiration for the design of these cost functions, are presented in the following section.

4.3 Cost Functions

One of the advantages of the proposed hierarchy formation framework is its independence from the design objectives. The algorithm can be adjusted to any set of requested objectives. The adaptation process is done through the utilization of metrics and the introduction of the appropriate cost functions. If the optimization of these cost functions is performed appropriately from the optimization algorithm, a hierarchical structure that meets the targeted objectives will be formed.

The cost functions have to be designed carefully with respect to the available metrics and the set of objectives. In this work various cost functions are presented, which can be categorized in two large classes with respect to the targeted objectives. Through these classes of cost functions the effectiveness of the selected optimization (SA) algorithm is demonstrated and the quality of the functions in accordance to the pre-specified objectives is validated. Namely, the two classes of cost functions are:

- Cluster characteristics based cost functions
- Node mobility based cost functions

The two defined categories of cost functions fall under the same lines of the taxonomy utilized to classify the metrics. Thus, the functions that constitute these two categories of cost functions depend on the metrics that define the corresponding class of metrics. The cost functions that belong into the cluster characteristics class represent objectives that are related with the topological characteristics of the generated clusters. Accordingly, the cost functions that define the node mobility class represent hierarchy generation objectives related to the network dynamics. The existence of these network dynamics is due to the mobility of the participating nodes. In depth description and presentation of the introduced cost functions along with their formal definition is provided in the following subsections.

4.3.1 Cluster characteristics based cost functions

The hierarchy design objectives represented from this class of cost functions are related to the characteristics of the physical structure of the generated clusters. These characteristics have to do with the size, the diameter and the number of input/output points to/from the cluster. The control of the physical characteristics of the clusters, affects the performance of the applied networking protocols. As it has been shown from various studies [38][39] the size and diameter of the network affects the performance of the applied protocols. The various protocols present different performance characteristics with respect to the varying size and diameter of the network, due to their functionality (i.e. use of flooding, number of messages exchanges, etc.). When hierarchy is introduced in the network, then these protocols will be applied per generated cluster. By knowing the performance characteristics of the applied protocol and for which network size and/or diameter the corresponding protocol achieves its best performance then by adjusting these characteristics of the generated clusters to the optimal values, the desired protocol performance could be achieved.

This category of cost functions can be further classified into two subcategories with respect to the number of metrics they involve. These two subcategories are:

- Single objective cost functions
- Multi-objective cost functions

The single objective subcategory includes the cost functions that are based on a single metric and the multi-objective subcategory includes those that involve combination of metrics. The former set targets the fulfillment of a single objective related to the physical characteristics of the generated clusters and equally the latter set aims on accomplishing a mixture of objectives.

In the following paragraphs, the both the single objective and the multi-objective cost functions that attempt the adjustment of the physical structure of the hierarchy are being introduced. Furthermore, the reasoning behind their design is explained.

The evaluation of the ability of the proposed cost functions to meet the pre-specified objectives is presented in a later section of this chapter.

Single objective cluster characteristics cost functions

The cost functions of this category have been designed to control the physical characteristics of the generated clusters. The metrics involved in the construction of these cost functions belong to the cluster information category and are selected with respect to the clustering objectives set. Namely, the objectives that trigger the cost functions of this class are:

- Balanced Size Clusters
- Balanced Size Cluster Diameters
- Minimization of Border Routers
- Optimal Cluster Size

A more detailed explanation of these objectives and the presentation of the corresponding cost functions that attempt to accomplish these objectives upon their optimization follow. Initially, a brief overview of the parameters that represent the metrics involved in the construction of these cost functions is provided.

Parameter	**Definition**
K	Number of generated clusters
C_i	Cluster *i*
$\lvert C_i \rvert$	Size of cluster *i*
d_{C_i}	Diameter of cluster *i*
BR_{C_i}	Number of border routers of cluster *i*
$\lvert C_i \rvert^*$	Optimal size for cluster *i*

Table 4.1. Representation of the metrics involved in the construction of cluster characteristics based cost functions

Balanced Size Clusters

The objective of this class' cost functions is the generation of balanced size clusters. The requirement for equal size clusters across the network can be beneficial in the functionality of many networking protocols. When each cluster consists of almost the same number of nodes, the distribution of network resources across the generated clusters is fair and better load balancing is achieved. Furthermore, as we have already mentioned, the applied networking protocols present preferences in terms of the network characteristics (including network size), so that they can achieve the desired performance characteristics. If instances of these protocols are going to be applied to each of the generated clusters, then each one of these clusters must present the preferred physical characteristics so that the performance of the applied protocol is optimal. In that case and specifically for preferences related to the cluster size, each cluster must have identical size if the same protocol is going to be applied. The following cost functions have been defined for the generation of such clusters with respect to the balanced size requirement:

Definition 4.11 (Cost Function for the Generation of Balanced Size Clusters): The optimization of the cost function:

$$J(C) = \min_{C}\left(Var\left(\left|C_1\right|^2, ..., \left|C_K\right|^2\right)\right) \qquad (4.11)$$

results in the generation of balanced size clusters (cluster with equivalent size with respect to their membership characteristics).

Definition 4.12 (Cost Function for the Generation of Balanced Size Clusters): The optimization of the cost function:

$$J(C) = \min_{C} \sum_{i-1}^{K} \left| C_i \right|^2 \tag{4.12}$$

results in the generation of balanced size clusters (cluster with equivalent size with respect to their membership characteristics).

<u>*Balanced Size Cluster Diameters*</u>

Similarly, the objective that is represented from this set of cost functions is related to the physical structure of the generated clusters. Specifically, the proposed cost functions attempt to accomplish the generation of clusters that present similar diameter with respect to hops, across the network. The significance of such an objective is similar to the importance of generating balanced size clusters. There are protocols whose desired performance favors specific requirements. Such requirements are related to the physical characteristics of the network among others. Apart from the network size, another physical characteristic is the network diameter. When hierarchy is present, then instead of applying a single instance of the protocol throughout the network, multiple instances of the same protocol can be applied – an instance per generated cluster. Since the requirements for the desired performance of the applied protocol have been known, if any of these requirements is related to the diameter of the network, the latter can be adjusted accordingly. Due to the application of the same protocol at each cluster, the recommended diameter must be the same at every generated cluster. The optimization of the following cost functions (4.13), (4.14) is expected to provide the desired clustering map that satisfies the balanced cluster diameters objective.

Definition 4.13 (Cost Function for the Generation of Balanced Diameter Clusters):

$$J(C) = \min_{C} \sum_{i-1}^{K} \left(d_{C_i}^{2} \right) \tag{4.13}$$

results in the generation of balanced diameter clusters (cluster with equivalent diameter characteristics).

Definition 4.14 (Cost Function for the Generation of Balanced Diameter Clusters):

$$J\left(C\right) = \min_{C} \left(Var\left(d_{C_1}^{2}, d_{C_2}^{2}, ..., d_{C_K}^{2} \right) \right) \tag{4.14}$$

results in the generation of balanced diameter clusters (cluster with equivalent diameter characteristics).

Optimal Size Clusters

The objective represented from the cost function of this category is similar to the balanced size clusters objective, but more general. Each generated cluster is treated separately in terms of its size requirements. This generalization of the objective of balanced size clusters comes with the extra constraint that the sum of the clusters size is equal with the network size. Apart from this constraint the generation of clusters with pre-specified requirements on their size (optimal size of the cluster), can be very useful in cases where different protocols are applied in the various clusters. If the favorable performance for each of these protocols is achieved in different cluster size, then the size of the corresponding cluster must be adjusted accordingly. The balanced size clusters objective cannot handle such scenarios. The cost function that formalizes the optimal size cluster objective is defined as:

Definition 4.15 (Cost Function for the Generation of Optimal Size Clusters): The optimization of the cost functions:

$$J(C) = \min_{C} \sum_{i-1}^{K} \left(|C_i| - |C_i|^{*} \right)^{2} \tag{4.15a}$$

$$J\left(C\right) = \min_{C} Var\left[\left(|C_1| - |C_1|^{*} \right)^{2}, \ldots, \left(|C_K| - |C_K|^{*} \right)^{2} \right] \tag{4.15b}$$

result in the generation of clusters with pre-specified requirement on their size (optimal cluster size).

Minimization of Border Routers

Along the same lines as the previously defined classes of cost functions, this one involves cost functions that control another aspect of the physical structure characteristics of the generated clusters. The objective that the introduced cost functions attempt to accomplish is related to the number of “entrances” (“exits”) to (from) the generated clusters. The building block of these functions is the metric related to the number of border routers per cluster. The objective is the generation of clusters with minimum number of border routers.

Definition 4.16 (Cost Function for the Generation of Clusters with Minimum Number of Border Routers):

$$J\left(C\right)=\min_{C}\sum_{i-1}^{K}BR_{C_i} \tag{4.16}$$

results in the generation of clusters with equivalent number of border routers.

<u>*Multi-objective cluster characteristics cost functions*</u>

The cost functions above aim on the satisfaction of a single hierarchy generation objective. Even though the ability of the algorithm to optimize these cost functions and generate the desired hierarchy is very important, sometimes is not sufficient for the performance improvement of the network. That is because the overall performance of the network depends on the efficient functionality of many protocols and is described from many parameters. The optimization with respect to one parameter may affect negatively any of the other network parameters, resulting in the degradation of the network performance. Sometimes the combined improvement of the network performance is required with respect to multiple parameters (multiple objectives). Taking advantage of the ability of SA to optimize complex cost functions, the above objectives can be combined into single cost functions for multi-objective optimization of the network. Such cost functions that portray multiple objectives have been proposed through this work. The following ones are representative of the cluster characteristics class since they are combining single objectives presented earlier.

Balanced Size Clusters and Minimization of Border Routers

The cost function that represents the complex objective, involves both the generation of balanced size clusters and the minimization of the border routers of the network. The reasoning of this combination of objectives is to achieve simultaneously the advantages resulting from the generation of balanced size clusters and the minimization of border routers as they were presented earlier in this section. The cost function that realizes this combination of objectives is:

$$J\left(C\right)=\min_{C}\left(Var\left(\left|C_1\right|^2,....,\left|C_K\right|^2\right)+10^3*\left(\sum_{i=1}^{K}BR_{C_i}\right)\right) \qquad (4.17)$$

where the definitions of parameters involved are provided in Table 4.1. The effect of the multi-objective cost function on the generated hierarchy compared to the corresponding single objective ones is described in the performance evaluation section. Also the ability of SA to optimize complex cost function is verified from the simulation results.

Balanced Diameter Clusters and Minimization of Border Routers

Similarly, with the previous cost function, this one represents a combination of hierarchy generation objectives. It belongs in the category of the clustering characteristics based cost functions since both of the targeted objectives involved are related with the structure of the generated clusters. These two objectives are the generation of balanced diameter clusters and the minimization of border routers. The advantages from the application of such a scheme are expected to be the combination of the advantages of each of the objectives separately. These advantages have been highlighted earlier. The combination of these objectives is represented from the following cost function:

$$J\left(C\right)=\min_{C}\left(Var\left(d^{2}\left(C_{1}\right),d^{2}\left(C_{2}\right),....,d^{2}\left(C_{K}\right)\right)+10^{2}*\left(\sum_{i=1}^{K}BR_{C_{i}}\right)\right) \qquad (4.18)$$

The optimization of cost function (3.13) from SA results in a hierarchical structure which complies with the objectives set.

- Optimal Cluster Size and Minimization of Border Routers

The complex objective represented from cost function (4.19) below, is a combination of two single objectives, the generation of optimal size clusters and the minimization of border routers. The effect of this type of hierarchy is expected to have the combined advantages provided from each one of the targeted objectives. The cost function that formalizes the combinational objective is:

$$J(C)=\min_{C}\left[\sum_{i=1}^{K}\left(\left|C_{i}\right|-\left|C_{i}\right|^{*}\right)^{2}+10*\sum_{i=1}^{K}BR_{C_{i}}\right] \qquad (4.19)$$

The verification of the ability of (4.19) to meet the complex objective is provided in the performance evaluation section of this chapter.

4.3.2 Node mobility characteristics based cost functions

In this category belong the cost functions that are related with the mobility characteristics of the generated clusters. The metrics that constitute these cost functions are those in the node mobility class. The targeted objective is related to the robustness of the generated hierarchical structure. The robustness is defined from the stability of the clusters' membership. The advantage of generating a hierarchical structure that remains stable in a dynamic environment is the minimization of the membership changes. The topology modifications due to nodes mobility impose extra overhead in maintaining the hierarchical structure. This overhead may be harmful to the overall network performance, so if we want to take advantage of the hierarchy benefits then it is preferable to eliminate it. A way to do it is to incorporate network environment characteristics into the hierarchy generation mechanism. Such characteristics are the mobility of the participating nodes. If we group together nodes that present similar mobility characteristics, then it is expected that these groups will remain connected for larger periods of time compared to those obtained without taking into account the similarities on the mobility patterns of the nodes. Adopting the philosophy behind the generation of robust to mobility clusters, we introduce the following cost functions.

The objectives represented from the cost functions of this class aim on the generation of robust to mobility clusters. For achieving the latter, the grouping of the nodes with similar mobility characteristics is attempted. Each of the introduced cost functions involves exactly one of the node-mobility metrics defined earlier. The parameters that represent these metrics in the definitions of the cost functions of this class are provided in Table 4.2:

Parameter	**Definition**
K	Number of generated clusters
C_i	Cluster *i*
$\left\|C_i\right\|$	Size of cluster *i*
$\theta_{r_{i,j}}$	Relative direction of nodes *i*, *j*
$U_{r_{i,j}}$	Relative Velocity of nodes *i*, *j*
LET_{ij}	Expiration Time of Link between nodes *i*, *j*
U_i	Scalar speed of node *i*

Table 4.2. Representation of the metrics involved in the construction of cluster characteristics based cost functions

The hierarchy generation objectives requested and the corresponding cost functions defined are:

Groups of Nodes with Similar Direction

Since the main objective of this class of cost functions is the generation of robust clusters, this can be achieved by grouping together nodes with similar mobility characteristics. These groups of nodes are expected to remain connected for longer periods of time, reducing the hierarchy maintenance overhead and increasing the effectiveness of the applied hierarchy. A cost function that attempts to meet the requested objective upon its optimization is defined as:

$$J\left(C\right)=\min_{C}\sum_{z=1}^{K}\left(\sum_{i,j=1}^{|C_z|}\theta_{r_{i,j}}\right)^2 \tag{4.20}$$

The mobility metric utilized for the introduction of this cost function is the relative direction of the nodes in a cluster. Even though the cost is affected from the relative direction of every pair of nodes in a cluster, independently from the existence of a direct link or not between them, the constraint of topological clusters is still imposed. The enforcement of the constraint is done during the generation of new clustering maps from SA. More details about the generation mechanism of the candidate clustering maps and its importance on the performance of the algorithm are provided in a later section of this chapter, where we describe the implementation.

Groups of Links with Similar Velocity

Similarly, with the previous cost function, the hierarchy generation objective represented from this cost function is the generation of robust to mobility clusters. In this case, instead of using the direction of the nodes as the mobility metric of choice, their velocity is used. Specifically, the cost function introduced involves the relative velocity of the nodes in a cluster and attempts the grouping of nodes with similar velocity characteristics. This cost function is defines as:

$$J\left(C\right)=\min_{C}\left(\sum_{z=1}^{K}\left[\sum_{i,j=1}^{|C_z|}U_{r_{i,j}^{z}}^{2}\right]^{2}\right) \tag{4.21}$$

The difference of (4.21) compared to (4.22) is that the latter involves only the direction of the nodes, as opposed to the former which is based on velocity, which is aware of both the speed and direction of the nodes. Thus, (4.22) is expected to be more accurate on the grouping of nodes with similar characteristics, resulting in more robust hierarchical structures. The latter expectation turns to be true, as it will be presented on the simulation analysis section of the introduced cost functions.

Groups of Links with Large LET

Another node mobility metric that is involved in the definition of cost functions that pursue the generation of robust hierarchy is the Link Expiration Time (LET). The cost function based on LET attempts to group together nodes so that the links they define are characterized from high LET. The higher is the LET value of a link, the longer is expected to remain on. The definition of metric is given by (4.11) and the corresponding cost function is formalized by (4.22), below.

$$J\left(C\right)=\min_{C}\left[-\sum_{z=1}^{K}\left(\sum_{i,j=1}^{|C_z|}\left(LET_{ij}\right)\right)^{2}\right] \tag{4.22}$$

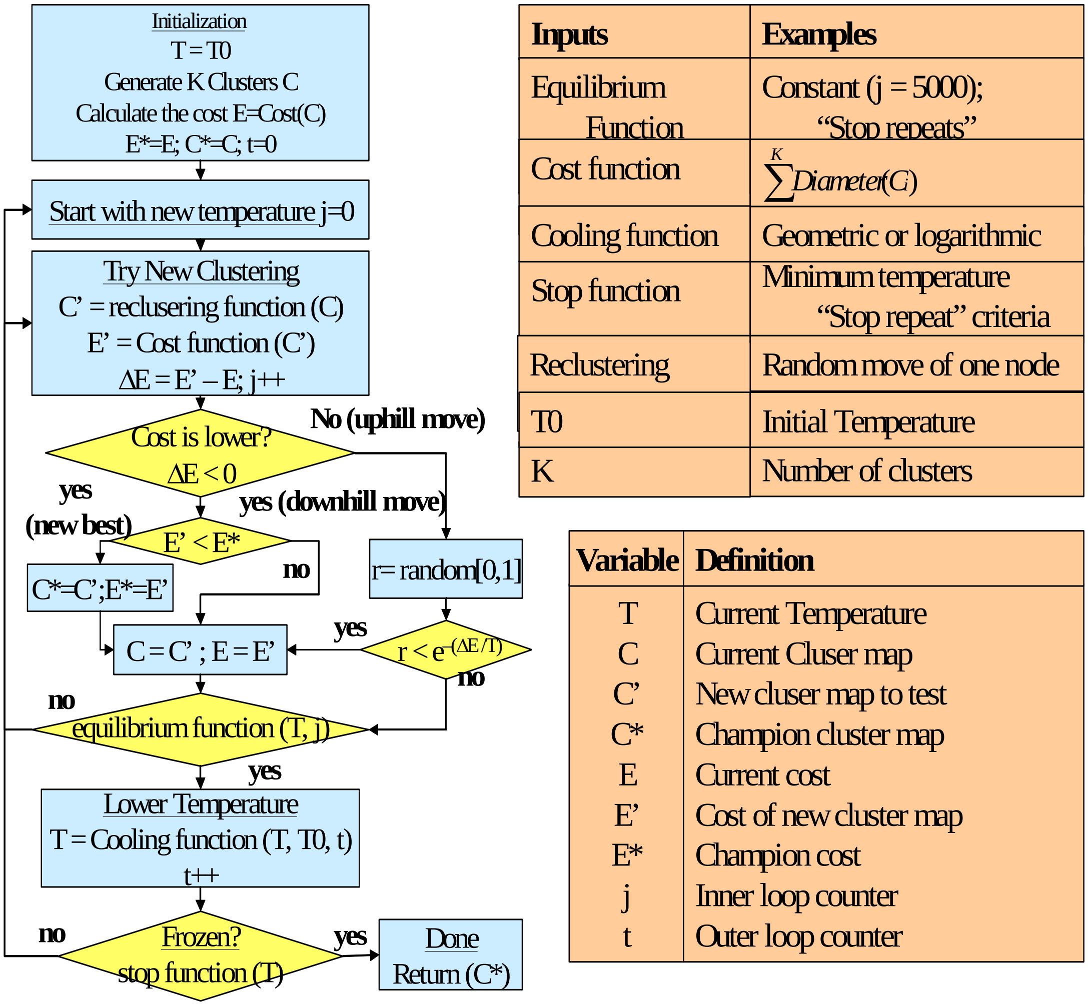
Since LET involves the direction and speed of the nodes that define the various links, (4.22) as (4.21) is aware of the combined mobility metrics of the nodes. It is expected to recognize more accurately and group more efficiently than (4.20), the nodes with similar mobility characteristics. Its performance with respect to (4.20) and (4.21) on the grouping of the nodes and the construction of robust hierarchical structures is provided in a later section in this chapter.

4.4 Performance Evaluation

Even though the algorithmic details of the hierarchy generation framework have been presented, its ability to fulfill the pre-specified objectives has not been shown yet. This ability is mostly related to the definition of the cost functions from the available metrics so that the targeted objectives are represented accurately. In this section the effectiveness of the introduced cost functions is demonstrated through a sequence of emulation results. The results were collected by optimizing the various cost functions using the modified SA algorithm and then the consistency of the generated clustering maps with respect to the targeted objectives was being checked. Prior to the presentation of the results, the configuration of the optimization algorithm (SA) which was used is provided in the next section.

4.4.1 Configuration of modified SA

The SA algorithm was configured with respect to the modifications and adjustments that were presented in the previous chapter. Even though the simulation analysis of the SA algorithm with these modifications and adjustments showed that the convergence time is significantly improved with small or no loss in the quality of the generated solution, the quality of the obtained solution was not studied for a larger set of cost functions (objectives). Before presenting the ability of the optimization algorithm to produce clustering map solutions in accordance to the pre-specified objectives, the configuration details of the SA algorithm utilized for the optimization of the corresponding cost functions will be given. These details refer to the selection of the cooling schedule, the *StopRepeat* number, the transition probabilities, the iterations per temperature, and the initial temperature. A brief overview of the rest modules of SA algorithm is also given. These modules are related to the generation of new feasible clustering maps per iteration and the evaluation of their cost. The algorithm that describes the functionality is given in figure 4.5.



Inputs	Examples
Equilibrium Function	Constant (j = 5000); "Stop repeats"
Cost function	$\sum^{K} Diameter(C)$
Cooling function	Geometric or logarithmic
Stop function	Minimum temperature "Stop repeat" criteria
Reclustering	Random move of one node
T0	Initial Temperature
K	Number of clusters

Variable	Definition
T	Current Temperature
C	Current Cluser map
C'	New cluser map to test
C*	Champion cluster map
E	Current cost
E'	Cost of new cluster map
E*	Champion cost
j	Inner loop counter
t	Outer loop counter

Figure 4.5. Simulated Annealing algorithm for network partitioning

Table 4.3 presents the values for the various parameters of the SA algorithm, as they were selected for the optimization of the various cost functions. These values will be justified in chapter 5, as part of the modification and adjustment of the optimization algorithm so that it can be applied in dynamic environments.

Parameters	**Configuration Values**
Cooling Schedule	Geometric Cooling Schedule
StopRepeats	100
State Transition Probabilities	Uniform
Initial Temperature	40
Iterations Per Temperature	100

Table 4.3. Configuration values of the SA parameters for the optimization of the introduced cost functions

Apart from the parameters, important role in the optimization algorithm is the generation of new feasible clustering maps and their cost evaluation modules. The generation of new feasible maps is performed as it will be described in chapter 5, based on the migration of one randomly selected node from a randomly selected cluster to a new randomly selected cluster. After this move is being performed, the old and new host clusters of the node are being checked for feasibility (topological clusters). If both are feasible the cost of the new clustering is evaluated with respect to the optimized cost function. If not then a new migration move is performed until a feasible clustering map is obtained. The cost evaluation of the feasible clustering maps in each round of SA is performed using the energy update method (see chapter 5), where the new cost is evaluated as an update from the previous cost by taking only into account the changes on the previous clustering map (e.g. currently optimal map).

The following section presents the ability of the algorithm to converge into hierarchical structures that satisfy the pre-specified generation objectives. Initially, the cost functions based on the cluster characteristics are optimized and then the node mobility cost functions are plugged into SA for the generation of optimal clustering maps.

4.4.2 Cluster characteristics based cost functions

The importance of the results to be presented here is twofold. Initially, to investigate the efficiency of the cost functions of the cluster characteristics based cost functions class and secondly to evaluate the ability of the adjusted SA algorithm to optimize these cost functions as fast as possible. The former is extremely critical, because a carefully designed cost function has to be exact on accomplishing the hierarchy generation objectives and at the same time has to be optimized comparatively faster than other cost functions that represent the same objectives. The evaluation of the introduced cost functions becomes even more interesting in the case of multi-objective cost functions. The efficiency of this type of cost functions is evaluated with respect to their ability to meet all the involved hierarchy generation objectives. The experimental results presented here for the introduced multi-objective cost functions are analyzed for efficiency. These results are highly correlated to the effectiveness of the optimization method in the case where multiple objectives have to be met and complex cost functions have to be optimized.

The evaluation of the cost functions was performed with respect to a large number of networks of different characteristics (size, degree of nodes) and number of generated clusters. Some representative graphs were selected to be demonstrated here, that capture the ability of the cost functions to accomplish the pre-specified objectives. These results were obtained for a network of 100 nodes that were dispensed randomly in an area of 500m x 500m. The average node degree $\overline{dgr_n}$ and the variance of the node degree $Var\left(dgr_n\right)$ are shown in the following table:

Statistics	**Values**
$\overline{dgr_n}$	5.78 neighbors/node
$Var\left(dgr_n\right)$	4.27

Table 4.4. Statistics of the network (fig. 4.7)

The density of the nodes per 10^4 m^2 is represented by the following 3-D graph:

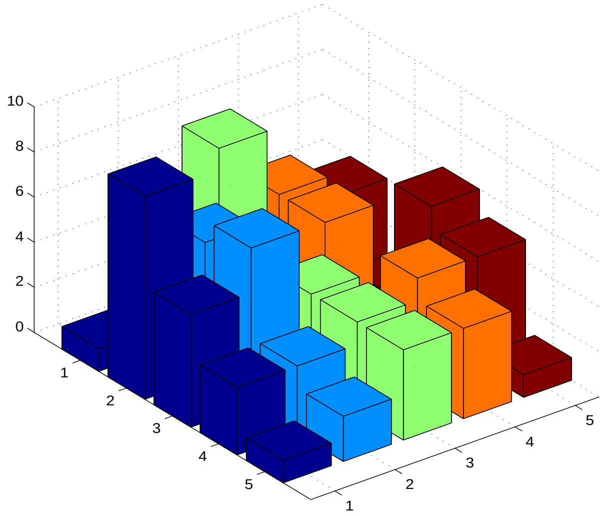


Figure 4.6. Density of nodes per 10^4 m^2

The topology and the connectivity of the nodes of this network are shown in figure 4.7.

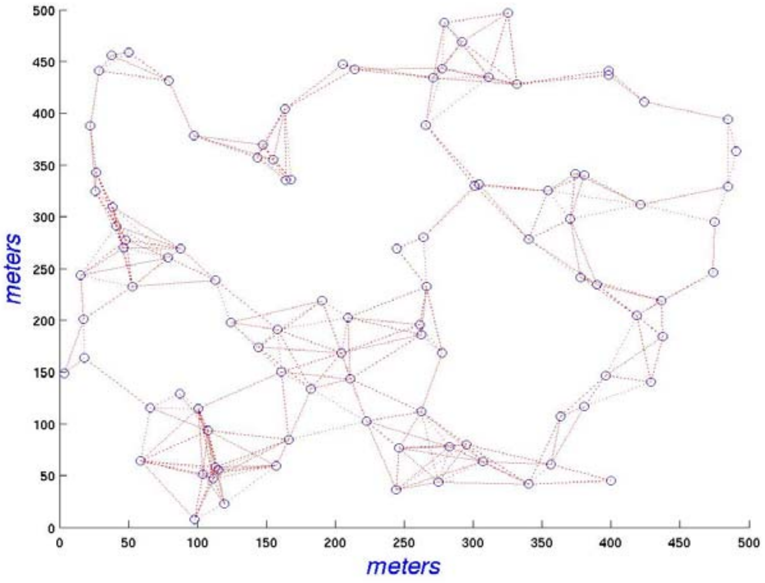


Figure 4.7. Network topology (100 nodes) of the demonstrated results

For the specific experiments performed for the collection of the demonstrated results the number of clusters generated was 5. Initially the results related to the single objective cost functions are presented and evaluated. Then the results associated with the multiple criteria cost functions are also demonstrated and analyzed.

4.4.2.1 Single Objective Cost Functions

The graphs for the introduced cluster information based cost functions are presented here. The following table recapitulates these functions and the hierarchy generation objectives that represent – the parameters are explained in table 4.2.

Cost Function	Objective
$J(C)=\min_{C}\left(Var\left(\lvert C_1\rvert^2,....,\lvert C_K\rvert^2\right)\right)$ $J(C)=\min_{C}\sum_{i-1}^{K}\lvert C_i\rvert^2$	Balanced Sized Clusters
$J\left(C\right)=\min_{C}\left(Var\left(d_{C_1}^2,d_{C_2}^2,....,d_{C_K}^2\right)\right)$ $J(C)=\min_{C}\sum_{i-1}^{K}\left(d_{C_i}^2\right)$	Balanced Diameter Clusters
$J(C)=\min_{C}\sum_{i-1}^{K}\left(\lvert C_i\rvert-\lvert C_i\rvert^*\right)^2$	Adjusting the Cluster Size
$J\left(C\right)=\min_{C}\sum_{i-1}^{K}BR_{C_i}$	Minimization of Border Routers

Table 4.5. Single Objective Cluster Information Based Cost Functions.

From the large scale experiments performed, the clustering maps generated upon the optimization of these cost functions were complying on the objectives set. Some indicative results related to this category of cost functions are:

- $J(C)=\min_{C}\left(Var\left(\lvert C_1\rvert^2,....,\lvert C_K\rvert^2\right)\right)$

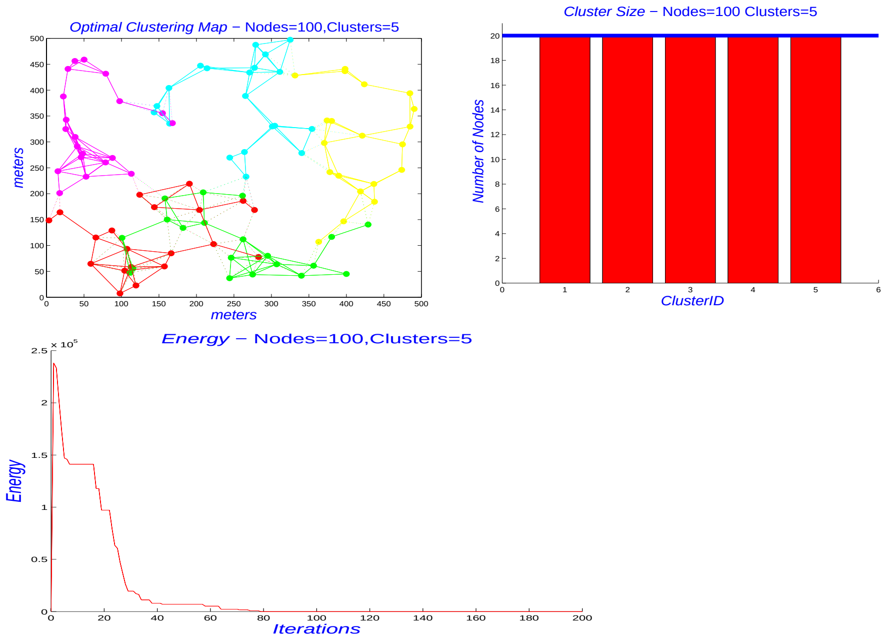


Figure 4.8. Balanced Size Clusters

Figure 4.8 demonstrates a group of results related to the generation of balanced size clusters and specifically to cost function (4.11). The upper left corner of the graph is the optimal clustering map outcome from SA, where the nodes that belong into the same cluster are marked with the same color. On the upper right corner each column corresponds to the cardinality of the generated clusters. This sub-graph presents the ability of the cost function to meet the pre-specified objectives. Obviously, (4.11) generates perfectly balanced size clusters upon its optimization (e.g., for 100 nodes network, 5 clusters of 20 nodes have been generated). The bottom sub-graph (energy vs. iterations) demonstrates the evolution of cost (energy) in each iteration of the optimization process. This result is an indication of the speed of the cost function in meeting the pre-specified objectives. Further analysis of the latter type of results is being performed later in the chapter.

- $J(C) = \min_{C}\left(Var\left(d_{C_1}^2, d_{C_2}^2, ..., d_{C_K}^2\right)\right)$

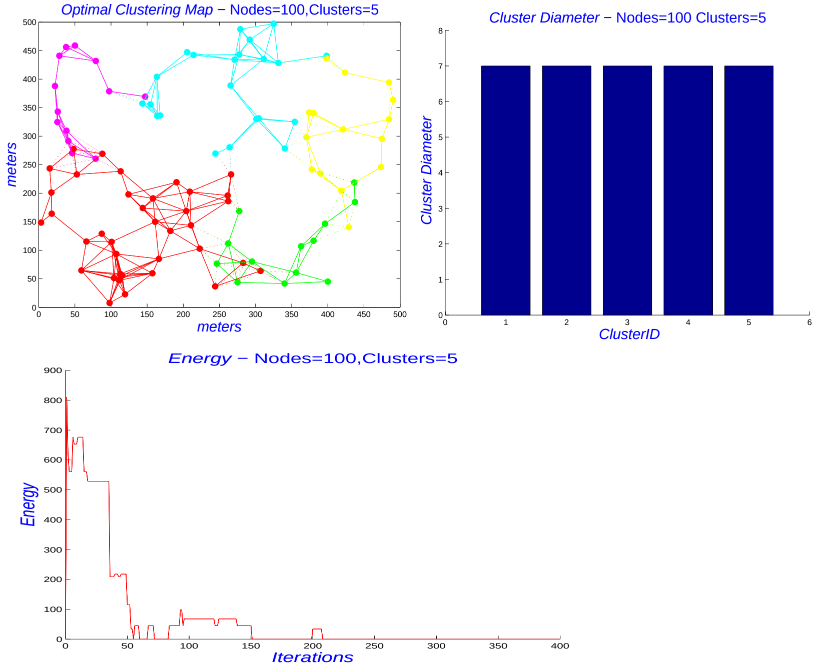


Figure 4.9. Balanced Diameter Clusters

In the same fashion as in figure 4.8, the above figure 4.9, involves a group of subgraphs related to the ability of the cost function (4.14) to generate balanced diameter clusters upon its optimization from SA. As it can be observed from the upper right corner subgraph, (4.14) is capable of meeting the hierarchy generation objectives that represents. In the specific experiment of figure 4.9, 5 clusters of diameter 7 hops each has been generated. Considering the large scale experimentation with this cost function and its optimization from SA, it appears to be very accurate on producing balanced diameter clusters and in small number of SA iterations (~400) as the bottom subgraph of the above figure demonstrates.

- $J(C) = \min_{C} \sum_{i-1}^{K} \left(|C_i| - |C_i|^* \right)^2$

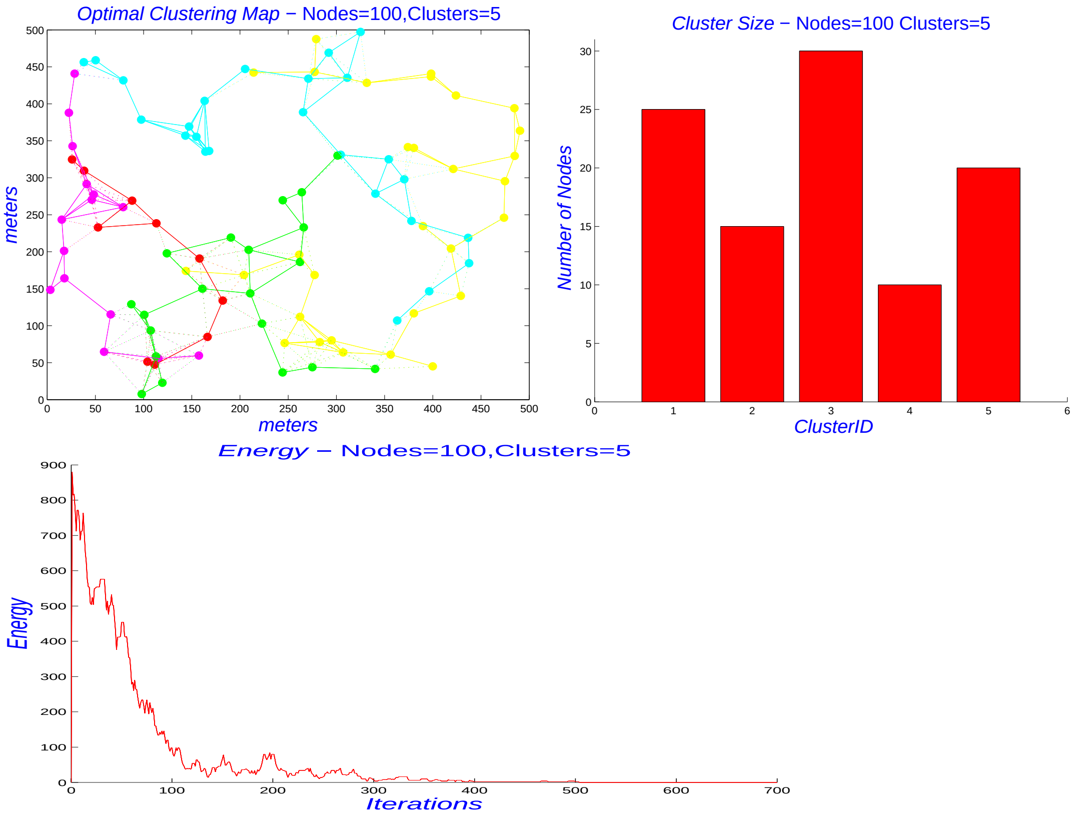


Figure 4.10. Optimal Cluster Size Assignments Cost Function

Figure 4.10 demonstrates results related to the optimal cluster size assignment cost function (4.15). For the specific experiment, the requested sample cardinalities for the 5 clusters to be generated were:

Optimal Cluster Size Assignments	25	15	30	20	10

Table 4.6. Requested cardinality for each generated cluster

The upper right subgraph represents the resulted cardinalities upon the optimization of (4.15). These cardinalities match perfectly the requested ones from Table 4.6, which is an indication of the ability of the specific cost function to accomplish the corresponding hierarchy generation objectives. Large scale experimentation with (4.15) has shown that it is systematic on converging to clustering maps that match the requested cardinalities.

4.4.2.2 Multiple Objectives Cost Functions

The resulting clustering maps related to single objective cluster information based cost functions, have demonstrated their ability to construct hierarchical structures that comply with the requested objective. By collecting similar groups of results related to the corresponding multiple objective cost functions, the ability of SA to optimize such functions is demonstrated. Also, the results presented in this subsection illustrate the effectiveness of these functions to meet the multiple pre-specified objectives that they represent. Before the presentation of the results, an overview table of the multiple objective cluster information based cost functions that have been introduced, is provided.

Cost Function	Objectives
$J(C)=\min_{C}\left(Var\left(\left\|C_1\right\|^2,....,\left\|C_K\right\|^2\right)+\left(10^3*\sum_{i=1}^{K}BR_{C_i}\right)\right)$	Balanced Size Clusters and Minimization of BRs
$J(C)=\min_{C}\begin{pmatrix}Var\left(d^2(C_1),d^2(C_2),....,d^2(C_K)\right)+\\ \left(10^2*\sum_{i=1}^{K}BR_{C_i}\right)\end{pmatrix}$	Balanced Diameter Clusters and Minimization of BRs
$J(C)=\min_{C}\left[\sum_{i=1}^{K}\left(\left\|C_i\right\|-\left\|C_i\right\|^*\right)^2+10*\sum_{i=1}^{K}BR_{C_i}\right]$	Optimal Size Assignment and Minimization of BRs
$J(C)=\min_{C}\begin{pmatrix}Var\left(\left\|C_1\right\|^2,....,\left\|C_K\right\|^2\right)+\left(5\times10^3*\sum_{i=1}^{K}BR_{C_i}\right)+\\ \left(10*Var\left(d^2(C_1),d^2(C_2),....,d^2(C_K)\right)\right)\end{pmatrix}$	Balanced Size Clusters, Balanced Diameter Clusters and Minimization of BRs

Table 4.7: Multiple Objectives Cluster Information Based Cost Functions.

The results that follow represent a small subset of the above cost functions but the conclusions drawn are representative for the performance of this category of cost functions, as it has been observed from the large number of experiments performed. The following results correspond to the cost functions for the generation of balanced size clusters along with the minimization of border routers (BRs) and for the generation of balanced size and diameter clusters along with the minimization of BRs, respectively.

- $J\left(C\right)=\min_{C}\left(Var\left(\left|C_1\right|^2,....,\left|C_K\right|^2\right)+\left(10^3*\sum_{i=1}^{K}BR_{C_i}\right)\right)$

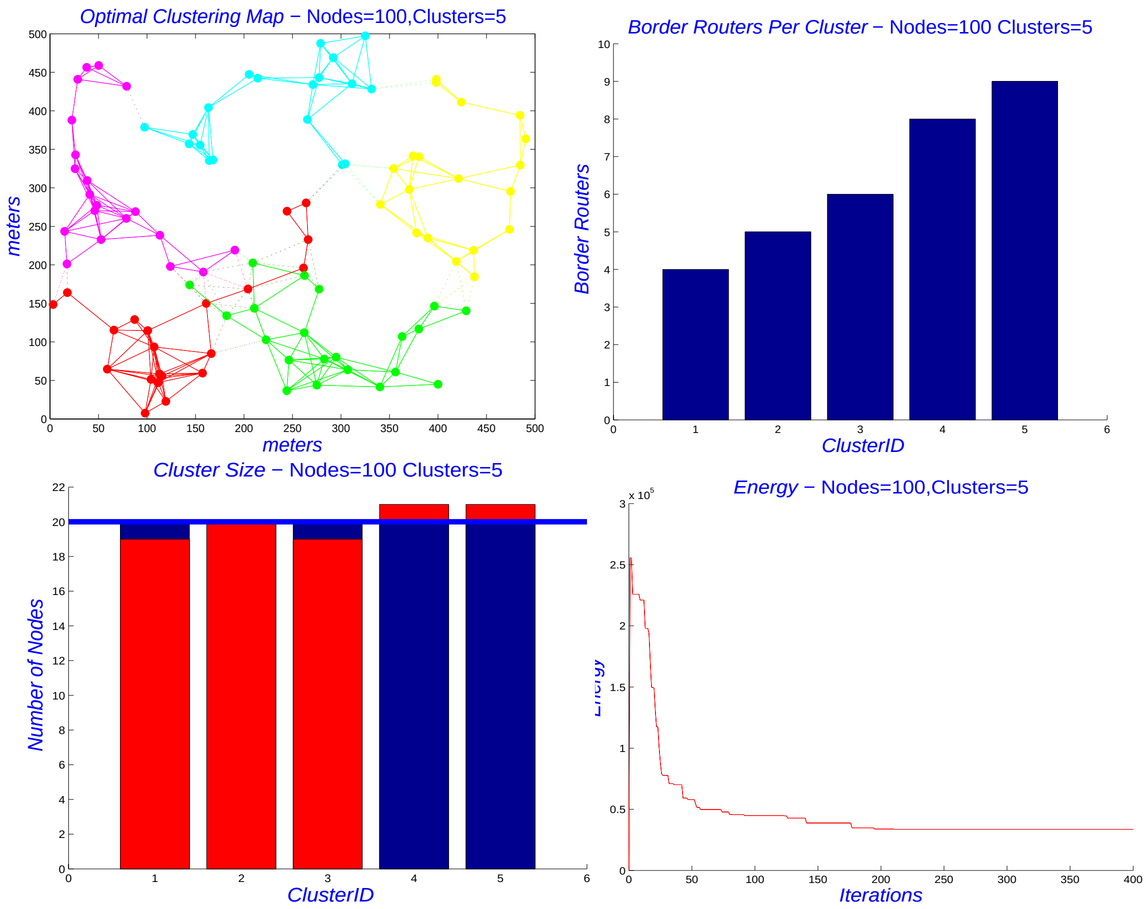


Figure 4.11. Multiple objectives cost function: Balanced Size Clusters and Minimization of Border Routers

A representative group of results for the two objectives cost function (4.17) is given in figure 4.11. The effect of adjusting the SA so that it is faster but it produces suboptimal solutions can be identified from the upper right and the bottom left subgraphs. The former provides the number of BRs per generated cluster and the latter demonstrates the cardinality of the generated clusters. About the cardinality subgraphs, the blue columns are the optimal values and the red represent the values obtained from the optimization. There is a small deviation from the optimal value but still the sizes of the generated clusters are very close to each other, almost balanced. When the minimization of BRs was the only targeted objective, the average number of BRs was ~30, with 26 the minimum observed. The same objective as part of a multi-objective cost function defined a slightly larger number of BRs in the network. For the specific experiment this number is 32. Due to the complexity of the cost function compared to the corresponding single objective ones, the solution obtained is suboptimal with respect to each objective individually. Whereas, the generated clustering map almost satisfies both of the objectives. By comparing the optimal clustering map of figure 4.8, which corresponds to the balanced size clusters, with this one, the effect of the multi-objective optimization is obvious. The generated clusters in the multi-objective case appear to be more isolated (e.g., less inter-cluster links). The observations made from the optimization of multi-objective cost functions illustrate the ability of adjusted SA to provide high quality clustering solutions very fast, even in complex situations. This observation becomes more noticeable in the following case where the optimization involves three objectives.

- $J(C)=\min_{C}\begin{pmatrix} Var\left(\left|C_1\right|^2,....,\left|C_K\right|^2\right)+\left(5\times10^3 * \sum_{i=1}^{K} BR_{C_i}\right)+ \\ \left(10*Var\left(d^2(C_1),d^2(C_2),....,d^2(C_K)\right)\right) \end{pmatrix}$

As with the previous cost function, this one involves more than one hierarchy generation objectives. Specifically, three are involved: the generation of balanced size clusters, balanced diameter clusters and the minimization of border routers. The collection of subgraphs demonstrating the effectiveness of this cost function on meeting all three objectives simultaneously is provided in figure 4.12. Similarly, the results obtained for each individual objective are suboptimal (e.g., slightly unbalanced cluster sizes and diameters and little higher number of border routers). This slight suboptimality is due to tuning of SA for its faster convergence, but as expected this has effect on the optimality of the generated solutions. The solution obtained is still of high quality especially for the dynamic networks under

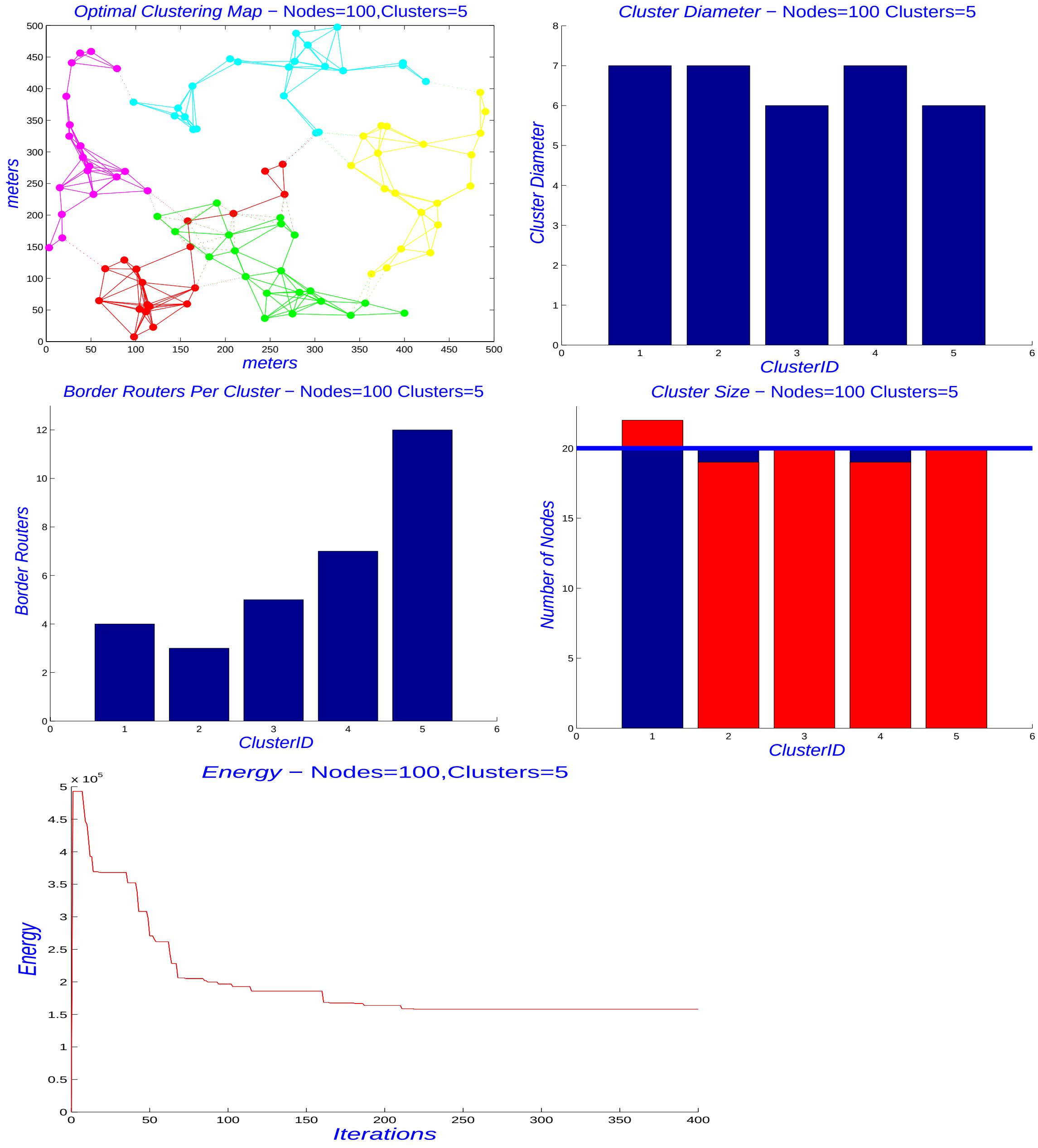


Figure 4.12. Multiple objectives cost function: Balanced Size Clusters, Balanced Diameter Clusters and Minimization of Border Routers

consideration, since the topology changes unavoidably will degrade the quality of the solution with respect to time. It is preferable to get a high quality suboptimal solution fast rather than getting an optimal which might require large amount of processing time. In dynamic environments the latter approach might construct hierarchical maps that are no longer feasible with respect to the topological constraints, if the solution has not been obtained fast enough compared to the network dynamics.

4.4.3 Node mobility characteristics based cost functions

The performance of this class of cost functions is being evaluated with respect to their ability to identify accurately groups of nodes that have similar mobility characteristics. The significance in forming clusters out of such groups of nodes is on the robustness of the generated hierarchy. These clusters are expected to maintain their membership stable for long periods of time due to the similar moving patterns of their members. The performance of the network will improve due to the reduction of the hierarchy maintenance overhead and the benefits provided from the application of the hierarchical structure.

The evaluation of the introduced node mobility cost functions involves their ability to identify different mobility groups of nodes, when these groups present different levels of distinctiveness (e.g., difference in speed and direction of movement) in their mobility characteristics. The experimental set up for the analysis of the proposed cost functions and the results collected are provided in the following sections.

4.4.3.1 Experimental Set Up

The effectiveness of the combination of SA with the proposed cost functions is evaluated by setting up carefully the experimental environment. The main element of the experimental set up is that the networks consist of predefined mobility groups with distinct mobility characteristics and the clustering framework had to identify these mobility groups as accurately as possible. The clustering framework applied for the identification of the mobility groups consists of the SA algorithm and the class of node mobility cost functions.

Since we are interested in group mobility we had to select the appropriate group mobility model. In our experiments we utilized the Reference Point Group Mobility (RPGM) Model. In RPGM we define a number of Reference Points (RPs) equal to the number of mobility groups we want to establish. To complete the definition of mobility groups, each node is assigned to a RP. The movement of the nodes is characterized from the mobility patterns of their corresponding RPs. These mobility patterns are assigned manually to the various RPs in the form of trajectories. When a RP moves to a new location each corresponding node is assigned to a random radius and direction around the new position of the RP. Because of the functionality of RPGM model and the randomness in the selection of the new node position, it is obvious that nodes that belong into the same group may have different speeds and directions, which makes the clustering of the various mobility groups more challenging but improves the significance of the experimental analysis of the proposed cost functions.

Two mobility groups were predefined by splitting the network of figure 4.11 into two topological clusters of same size (50 nodes each). The RP for each of the groups was selected as the center of gravity of their members coordinates. The trajectory of these points defines the new position of the corresponding group nodes.

The RP trajectories were predefined so that the RPs where moving on a straight line with constant relative direction θ_{RP_1,RP_2} and constant relative speed $S_{RP1,RP2}$, as it is indicated in figure 4.13.

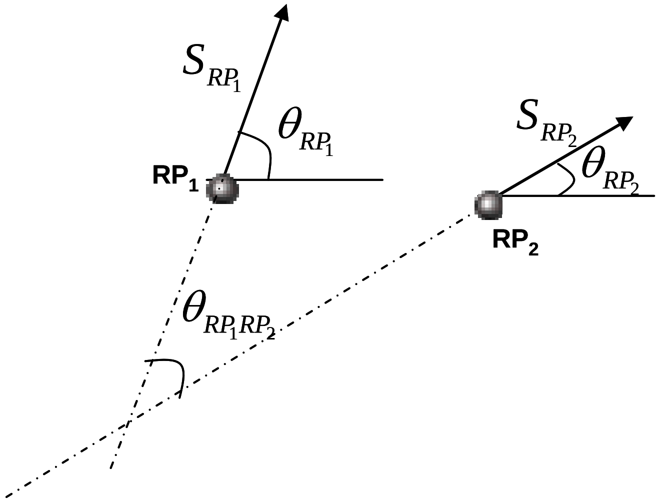


Figure 4.13. Experimental set up: Based on the RPGM model, 2 mobility groups are defined with respect to RP_1 and RP_2

We varied the relative direction from 0^o to 360^o (e.g., $\theta_{RP_1,RP_2} \in \left[0^o...360^o\right]$) by a step of 15^o. We repeated each step 100 times. For each run we measured the percentage (%) of nodes that they were assigned in a group different than the one they were pre-assigned to. The average percentage of incorrect assignments of 100 runs for each θ_{RP_1,RP_2} is provided in figures 4.14 and 4.15 for the introduced node mobility based cost functions (4.20) and (4.21) respectively. Furthermore, figure 4.18 illustrates also the effect of relative speed $S_{RP1,RP2}$ of RP_1 and RP_2 on the accuracy of the cost function (4.21). The effect of $S_{RP1,RP2}$ is investigated only for the latter cost function since it involves this metric, as opposed to the former (4.20) cost function which involves only the relative direction of the participating nodes.

Figure 4.14 indicates that the cost function (4.20) can identify accurately the various mobility groups especially when the groups are moving in relative directions such that $\theta_{RP_1,RP_2} \in \left[30^o ... 330^o\right]$. When $\theta_{RP_1,RP_2} \notin \left[30^o ... 330^o\right]$ then the proposed cost function has

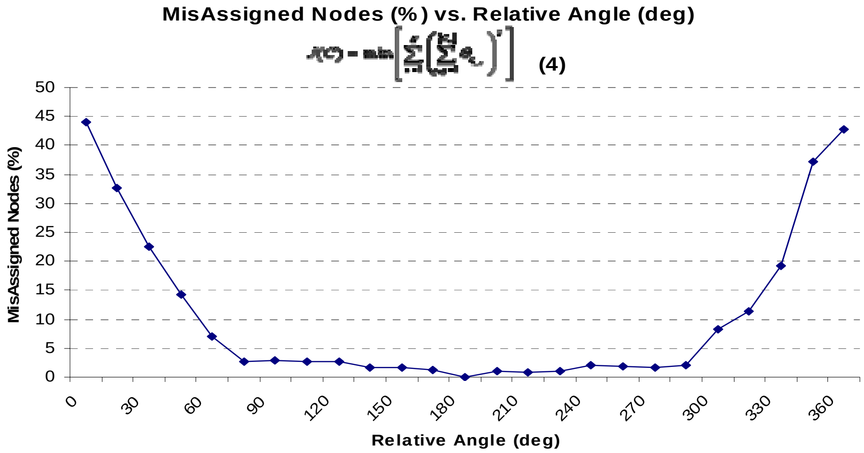


Figure 4.14. Incorrectly assigned nodes percentage (%) with respect to relative angle θ_{RP_1,RP_2} for cost function (4.20)

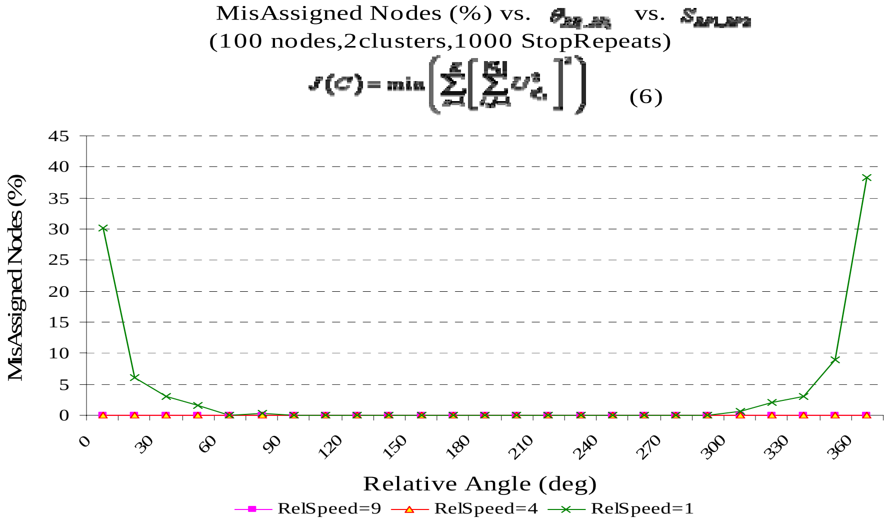


Figure 4.15. Incorrectly assigned nodes percentage (%) with respect to relative angle θ_{RP_1,RP_2} for cost function (4.21) for various relative speeds $S_{RP1,RP2}$.

difficulty on accurately identifying the pre-specified the mobility groups. This is not a limitation of the accuracy of the cost function since in this scenario the accurate selection of mobility groups is not restricted to the original mobility groups, because of the similarity in their directions. But still we can do better if we incorporate the speed of the participating nodes into the cost function. By doing so, figure 4.15 indicates that cost function (4.21) presents much better accuracy than cost function (4.20) which depends solely on the nodes direction. For $S_{RP1,RP2} > 1\ m/s$ the mobility groups are identified with accuracy 100%. The latter illustrates the effectiveness of the cost function (4.21) and the optimality of the decisions taken from SA algorithm.

4.5 Importance of Cost Function Selection

The same objective can be represented from various cost functions. Upon the optimization of the latter, the obtained clustering map may satisfy equally well the pre-specified objective. Such cost functions have been introduced earlier for the generation of balanced size clusters (4.11, 4.12), balanced diameter clusters (4.13, 4.14) and the optimal cluster size assignments (4.15a, 4.15b). Even though multiple cost functions exist for the same objective, only one can be applied. Further evaluation of these cost functions has to be made to determine which of these are the most appropriate to represent the corresponding objectives.

Since it has been shown that satisfy equally well the targeted objectives, the selection metric has to be something different but of equivalent importance. Such a metric is the speed of convergence of the SA algorithm. As it is shown in figure 4.16,

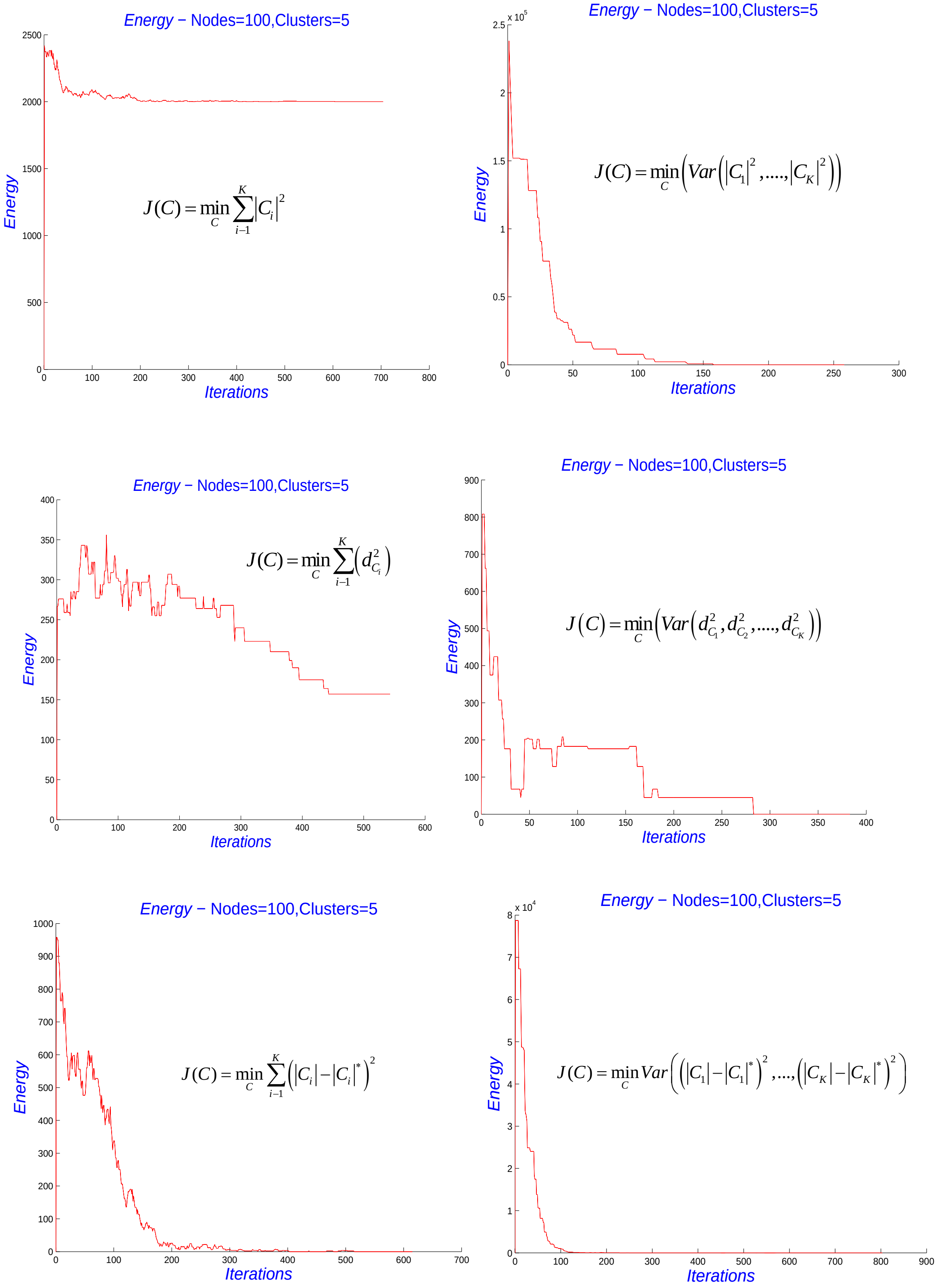


Figure 4.16. Energy behavior per iteration with respect to the cost function selection

the behavior of the cost functions that represent the same objectives result into different convergence speeds and energy behaviors when are optimized from the SA algorithm. The evaluation of the energy behavior and the convergence characteristics of SA with respect to equivalent cost functions, aims on the selection of the most efficient cost function.

The general forms of the contender cost functions are:

- Sum of squares: $\sum(\ )^2$
- Variance of squares: $Var\left((\ )^2,\ldots,(\ )^2\right)$

Figure 4.17 renders two major observations about these contender cost functions. The first observation is related to the required SA iterations until convergence and the second to the degradation rate of the energy with respect to the number of iterations.

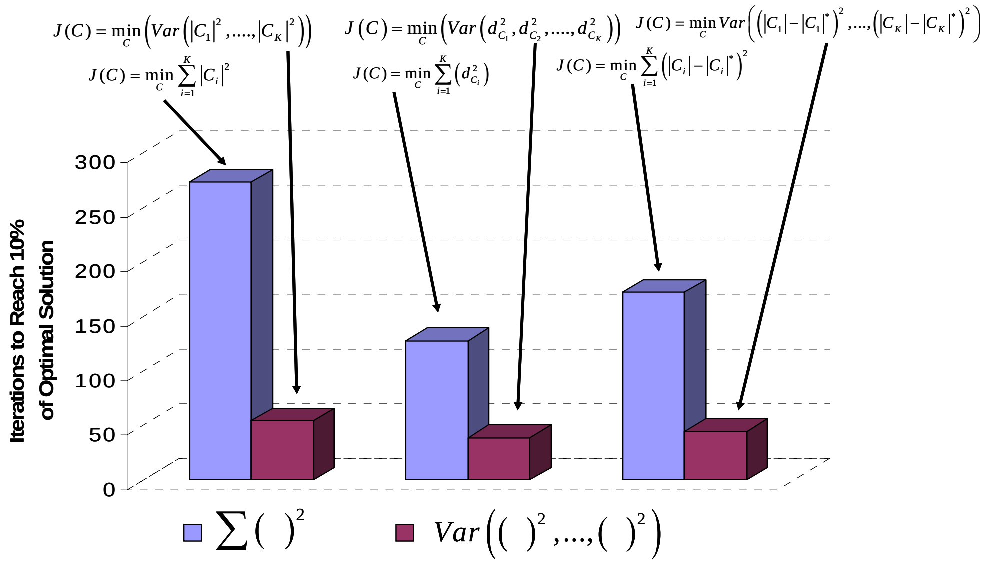


Figure 4.17. Average number of iterations required for reaching a solution 10% worse than the optimal

With respect to the degradation rate, the variance of squares based cost functions present better behavior, since the energy approaches the optimal value faster than the sum of squares based cost functions. The significance of this behavior is that in cases of early termination of the optimization process, the suboptimal solution obtained utilizing $Var\left((\ )^2,...,(\ )^2\right)$ will be better than the one obtained using $\sum(\ )^2$ for the same number of iterations. Figure 4.17 presents the average number of iterations required for SA to reach a solution that is 10% worse than the optimal, for each one of the contender cost functions. With respect to this indicative graph, $Var\left((\ )^2,...,(\ )^2\right)$ outperforms $\sum(\ )^2$, since for much less iterations in average, a better solution is obtained.

About the convergence speed of the contender cost functions, a more conclusive picture can be constructed from figure 4.18 where the average number of required iterations is provided. These results were collected from 100 SA applications on each of the cost functions.

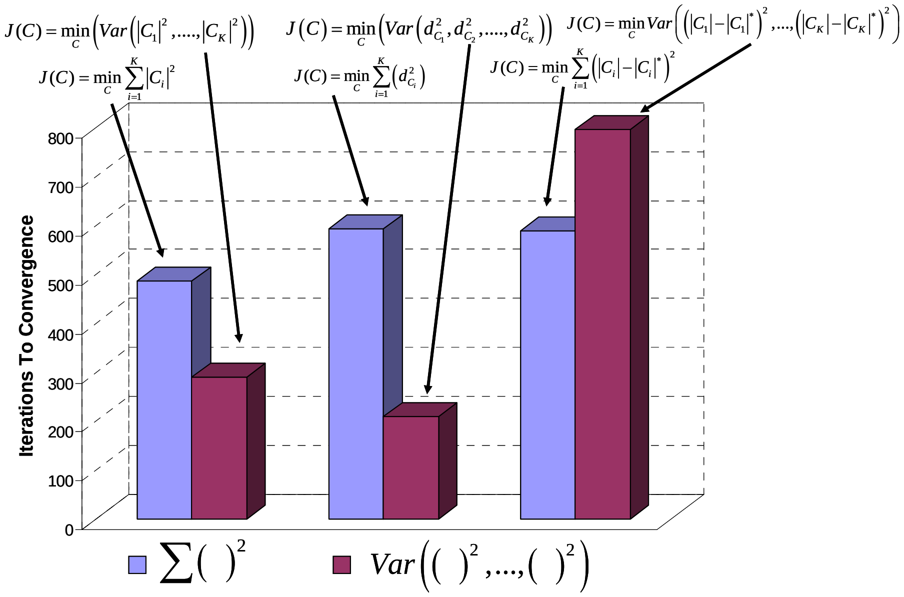


Figure 4.18. Average number of iterations required for convergence with respect to the cost function selection

$Var\left((\ )^2,\ldots,(\ )^2\right)$ illustrates more promising characteristics in terms of the required convergence iterations compared to $\sum(\ )^2$. For the majority cost functions convergences in fewer SA iterations. In two of the cases (balanced size clusters, balanced diameter clusters) examined here, $Var\left((\ )^2,\ldots,(\ )^2\right)$ appears to converge much faster – in 50% to 70% less iterations. For the optimal size assignments to clusters, $Var\left((\ )^2,\ldots,(\ )^2\right)$ appears to be slightly slower.

By combining the conclusions for the energy behavior and the convergence iterations, $Var\left((\ )^2,\ldots,(\ )^2\right)$ seems to have more promising performance, even though in special cases, $\sum(\ )^2$ has been shown to be the winner. It is important to select the cost function that speeds up the SA algorithm, satisfies the pre-specified objectives and presents the best energy degradation rate. Comparative study of the introduced cost functions is required to determine their characteristics, so that they can be applied accordingly.

4.6 Conclusions

This chapter presents the metrics considered, a sample set of hierarchy generation objectives and the corresponding cost functions introduced as part of the proposed clustering framework. The metrics considered are values related to the characteristics of the clusters to be generated, and the mobility of the participating nodes. An important attribute is that the values of the metrics can be measured in real time from the network. These metrics are the building blocks of the cost functions that represent the hierarchy generation objectives, which could be indicated from the current network conditions or be pre-specified from a network coordinator. A set of indicative cost functions has been introduced with respect to a set of objectives. These objectives are related to the characteristics of the clusters generated (e.g., size, diameter, number of BRs) or the mobility of the participating nodes (e.g. similar directions, similar velocity).

The evaluation of the cost functions introduced was performed with respect to their ability to satisfy the objectives they represent and their effect on the convergence time of SA algorithm. For the former as it has been indicated from the extensive experimental results, the cost functions introduced are very accurate on satisfying the targeted objectives, even for the cases where multiple (combination of) objectives are involved. This is also due to the ability of SA to perform successfully the optimization of very complicated cost functions. In the case of multiple objectives optimization, extra care must be taken for the assignment of weights to the various involved objectives in the cost function.

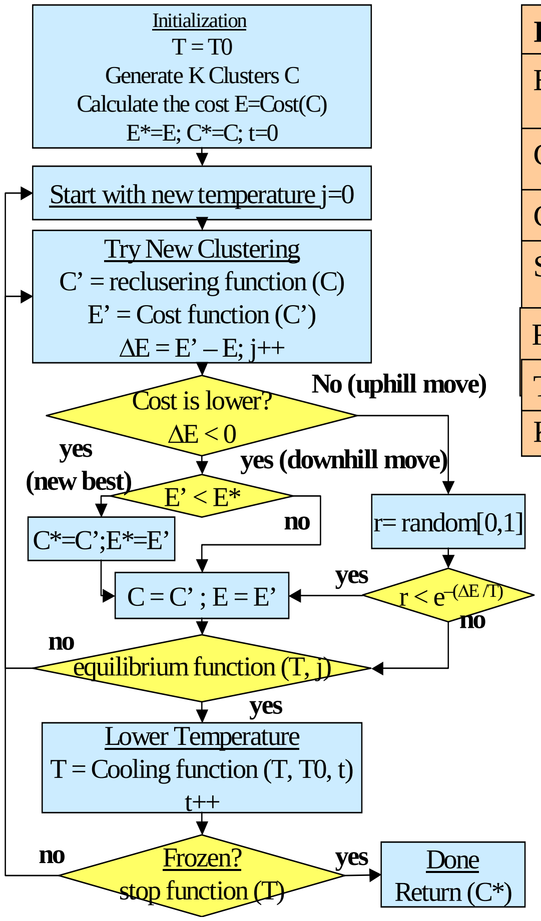
Last but not least, a very important observation is the significance of selecting the appropriate cost function among those that represent the same objectives. This is due to the effect the different functions may have on the convergence speed of SA, even though similar objectives are accomplished. Since speed of convergence is crucial for the effectiveness of the proposed clustering framework, special care must be taken to select the appropriate cost function. A representative case appeared among the set of the introduced cost functions. The evaluation between the functions consisting of the variance of squares and those involving the sum of squares, indicated that the variance of squares is the preferable option for accomplishing the pre-specified objectives while also the speed of convergence improves.

Chapter 5: Customizing Simulated Annealing (SA) for Dynamic Environments

5.1 Introduction

The design objective is to speed up the convergence of the algorithm. During this process, it is expected that the solution provided from the algorithm will not be the globally optimal. The more we speed up the algorithm, by forcing it to converge faster, the smaller is the surface of the solutions it explores and the lower the quality of the obtained solutions. Since the main objective of this work is the generation of hierarchical structures capable of improving the performance of the network, the algorithmic framework will be incapable of achieving this objective if the quality of the obtained solutions is sacrificed for the speed of convergence. Ideally, someone would prefer to obtain from SA the globally optimal solution as fast as possible. In general, due to the functionality of the algorithm, this cannot be guaranteed, so we have to trade off the optimality with speed of convergence. On the other hand, the adjustments of the various parameters and the modifications of the SA modules have to be performed such that the optimality of the solutions provided is not totally sacrificed for improving the speed of the algorithm. Still, high quality solutions are required so that the hierarchy generated satisfies the pre-specified set of clustering objectives.

The speed of convergence is more important for the network environment we consider, since the topology will be changing in accordance to the network dynamics (i.e. mobility and failures of the participating nodes, interference on the links). In such scenarios even if the globally optimal solution is obtained, after some time it may not anymore be sufficient for the network performance improvement. For that reason rapid convergence is more preferable rather than optimality. The various parameters and modules have been tuned in accordance to these design objectives. For adjusting and modifying the algorithm, the basic SA functionality was implemented with respect to figure 3.4 provided in chapter 3. The block diagram of the algorithm implemented is shown in figure 5.1. The details on the tuning of the parameters and modules of SA are given in the following subsections.



Inputs	Examples
Equilibrium unction	Constant (j = 5000); "Stop repeats"
Cost function	$\sum^{K} Diameter(C)$
Cooling function	Geometric or logarithmic
Stop function	Minimum temperature "Stop repeat" criteria
Reclustering	Random move of one node
T0	Initial Temperature
K	Number of clusters

Variable	Definition
T	Current Temperature
C	Current Cluser map
C'	New cluser map to test
C*	Champion cluster map
E	Current cost
E'	Cost of new cluster map
E*	Champion cost
j	Inner loop counter
t	Outer loop counter

Figure 5.1. Flow Diagram for the Implemented Simulated Annealing algorithm for network partitioning

5.2 Simulated Annealing: Tunable Parameters

In the simple nature of annealing, there lies the challenge in constructing efficient and effective implementations of the algorithm. There are many algorithmic parameters that have to be adjusted appropriately for obtaining the desired performance from the optimization algorithm. The ability to adjust the algorithmic parameters provides flexibility to the users so that they can configure the characteristics of the SA algorithm, in accordance to their performance preferences. On the other hand, this flexibility may be disadvantageous, since it requires a lot of effort to configure the algorithm appropriately for the solution of various classes of optimization problems. In this section we briefly present the tunable parameters and modules of SA algorithm, which have been configured appropriately in order the desired performance of the algorithm to be achieved.

The performance of the SA algorithm can lie between two extremes, which are controlled from the appropriate tuning of its parameters. These extremes are

a) Global optimality with the risk of large convergence times.

b) Speed of convergence with the risk of obtaining suboptimal solutions

The parameters of the algorithm that control the performance characteristics of the algorithm and require configuration are:

- **Cooling Schedule:** How the temperature (control parameter) decreases from an initial value towards a pre-specified final value or until the stop criterion is satisfied.
- **Cooling Factor:** In combination with the cooling schedule, determines the speed at which the temperature (control parameter) decreases.

- **Initial Temperature:** The initial value of the control parameter. This value has to be sufficiently high so that almost all transitions are accepted.
- **Termination Condition:** Determines the criterion for the algorithm to converge (stop criterion)
- **State Transition Probabilities:** How the new solutions are generated from the existing ones (generation mechanism)
- **Initial Solution:** The initial clustering map that is fed to the SA algorithm for its bootstrapping.
- **Length Plateau:** The number of iterations at every value of the temperature (control parameter). The value of this parameter has to be sufficiently large so that the stationary distribution holds (equilibrium) for every value of the control parameter (temperature).
- **Energy (Cost) Updates:** How the cost of the newly generated clustering maps is obtained in each iteration of the SA algorithm.

Due to the dynamics of the network environment under consideration, the objective is to tune the SA algorithm, so that optimality is trade off with speed of convergence. Furthermore, the provided solutions must still be of high quality (low cost) with respect to the hierarchy generation objectives. The following subsections, describe the tuning of the above parameters, so that the design objectives for the introduced hierarchy generation algorithmic framework are satisfied.

5.3 Customizing Simulated Annealing (SA) for Dynamic Environments

In this section the configuration of the SA algorithm is presented. The configuration is done through the adjustment of the various parameters of the algorithm so that a balanced trade off is maintained between the convergence time and the quality of the solutions obtained.

5.3.1 Termination Condition (Stop Criterion)

One of the most important parameters for speeding up the convergence time characteristics of the algorithm is the convergence criterion to be applied. The appropriate selection of this criterion will determine the ability of SA algorithm to terminate quickly but also to converge in clustering solution, which satisfies the hierarchy generation objectives. It is crucial to configure appropriately the termination condition so that the trade off between the convergence time and the optimality of the obtained solution is balanced. Our objective is to adjust the SA algorithm so that it can converge in real time on a high quality hierarchical structure – lowest (highest) cost solution possible in the case of minimization (maximization) optimization problem.

In theory the algorithm terminates when the temperature (control parameter) becomes zero. The algorithm at that point converges to the global optimal solution (reaches the global equilibrium in terms of the physical annealing process) nd for that reason is considered asymptotically as an optimization algorithm. In practice this will take an infinite number of iterations, so does not suite the practical implementation of the algorithm. Due to the latter, in practical implementations the algorithm is considered as an approximation algorithm. The method utilized as the termination condition of the algorithm is related to the improvement of the cost achieved by the subsequent iterations. Specifically the general practical termination condition can be defined as follows:

Definition 5.1 (Termination Condition): If the cost of the optimal solution obtained at the k^{th} iteration is C_k^*, which is more than $\varepsilon\%$ better compared to the optimal solution C_{k-1}^* of the $\left(k-1\right)^{th}$ iteration, then the algorithm terminates if in the $\left(k+n\right)^{th}$ iteration the optimal solution C_{k+n}^* as not improve the cost more than $\varepsilon\%$.

From the above definition, the termination condition is precisely formalized when specific values for the parameters ε and n are provided. These values depend on the required optimality of the solution and on the convergence time of the algorithm, as the study performed shows. In this study several of the cost functions introduced in the previous chapter were involved. The objective was to understand the trade off between optimality of the obtained solution and the resulting convergence time with respect to ε and n parameters. Since the objective was to speed up considerably the convergence time of SA but in such degree where the solutions obtained satisfy the pre-specified hierarchical generation objectives, ε was selected to be equal to 0. This design decision along with the selection of a small number of subsequent iterations n results in forcing the algorithm to converge faster without considerable degradation on the optimal solution obtained. This observation is highlighted also from the collection of results presented in the following graphs, where the optimality of the solutions obtained with respect to the selection of parameter n is provided. The version of the algorithm utilized for the collection of the results of this study is described from the block diagram of figure 5.1. The following graphs are for networks of 100 nodes where 5 domains were generated.

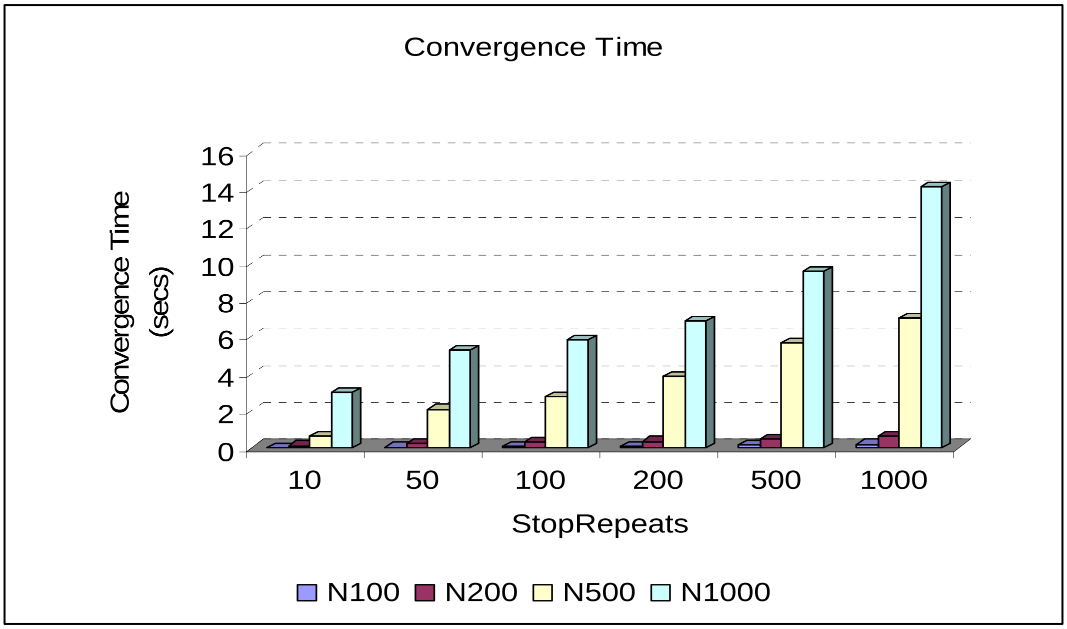


Figure 5.2. Convergence Time vs. stop-repeats (n)

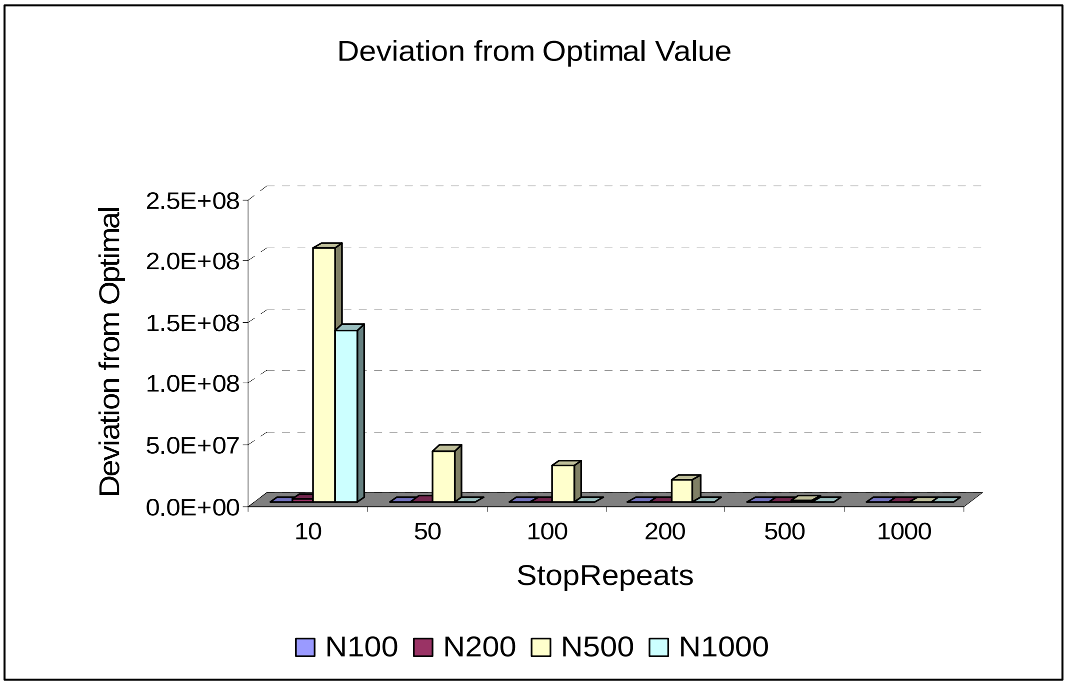


Figure 5.3. Deviation from optimal value with respect to the number of stop-repeats (n)

Figure 5.2 presents the convergence time of the algorithm for different values of n. The larger this number becomes the longer it takes to the algorithm to converge. This is expected since for large n the algorithm has to perform many more iterations to meet the termination condition. On the other hand the smaller the n, the higher the probability to obtain solutions that do not meet the hierarchy generation objectives (low quality clustering solutions). Figure 5.3 provides a representative snapshot of the solution's suboptimality for different values of n. The larger the value of the cost, the lower the quality of the clustering solutions obtained, with respect to the minimization problems introduced in the previous chapter. Obviously, the larger the n, the more optimal the solution.

The selection of n depends both on the quality of the clustering solution expected and on the dynamics of the network environment. The latter is the regulator factor since it may limit the quality of the achievable clustering solutions. For example if the topology of the network is fast changing then it may not be possible for the algorithm to obtain clustering solutions that satisfy the hierarchy generation objectives. On the other hand if the network remains unchanged for long periods of time then the SA algorithm can be applied with very large n so solutions close to the globally optimal ones can be obtained.

5.3.2 Cooling Schedule and Cooling Factor

Apart from the termination condition, an equally important parameter for the efficient functionality of the SA algorithm is the cooling schedule. As we have mentioned in chapter 3, the cooling schedule defines the rate at which the control parameter (temperature) is lowered. The importance of this rate lies on how effectively the solution space will be explored for the best solution. The effectiveness of traversing the solution space is linked on the Metropolis criterion (5.1) and specifically on the part of the criterion for the acceptance of deteriorations in cost instead of improvements only.

$$P_{c_{t+1}}\left(C^{*} \leftarrow C_{t+1}\right)=\begin{cases}1 & \text{if } \Delta J>0 \\ \exp\left(\dfrac{\Delta J}{c}\right) & \text{if } \Delta J \leq 0\end{cases} \tag{5.1}$$

The Metropolis criterion determines if the clustering solution C_{t+1} obtained in the $\left(t+1\right)^{th}$ will be able to substitute the currently optimal solution C^{*} given that the cost difference ΔJ, with respect to cost function J, between the two solutions is $\Delta J = J\left(C^{*}\right) - J(C_{t+1})$ and the value of the control parameter is c.

The larger the value of the control parameter the higher the probability

The relation of the control parameter c with the efficient traversing of the solution space is explained from the Metropolis criterion (5.1). The larger the value of the control parameter c, the higher is the probability that the algorithm will accept deteriorations in cost. Evidently, the longer the algorithm iterates for large values of the control parameter the better the surface of solutions is traversed since temporarily worse solutions may lead to better final solutions (e.g. avoidance of low quality locally optimal solutions).

On the other hand the longer the large values of c are maintained the less possible is the algorithm to converge quickly, since new clustering solutions will be accepted all the time, so the termination condition will be difficult to be satisfied. Even if the algorithm converges, the quality of the solution obtained will be questionable. The design objective indicates the utilization of a cooling schedule such that the rate of control parameter’s decrease is neither very slow (better traversing of the solutions’ surface but very slow for real time application of the algorithm) nor very fast (running into the risk of converging to very low quality clustering solution).

The asymptotic convergence of the SA algorithm was proven with respect to the logarithmic cooling schedule.

Definition 5.2 (Logarithmic Cooling Schedule): When the decrease rate of the control parameter (temperature) c follows the logarithmic cooling schedule, then the value of c is determined from the following function:

$$c_t = \frac{c_0}{1+\ln t} \tag{5.2}$$

where c_t and c_0 are the current and initial values of the control parameter, respectively and t specifies the number of iterations.

Even though the effectiveness of the logarithmic cooling schedule is proven, it is very slow in practice, so it is prohibitive with respect to the design objectives of this work. In order the SA algorithm to converge in real time another cooling schedule is required, so that the algorithm is faster but also the solutions obtained are of high quality. A cooling schedule that meets these requirements is the geometric cooling schedule which was adopted for the specific realization of the algorithm. The geometric cooling schedule is defined as follows.

Definition 5.3 (Geometric Cooling Schedule): When the decrease rate of the control parameter (temperature) c follows the geometric cooling schedule, then the value of c is determined from the following function:

$$c_t = a^t \cdot c_0$$

where c_t and c_0 are the current and initial values of the control parameter, respectively, t specifies the number of iterations and α $\left(0 < \alpha < 1\right)$ is the cooling factor, which determines the decrease rate.

Oppositely to the logarithmic cooling schedule, the effectiveness of the geometric cooling schedule has not been proved theoretically. Many experimental studies have indicated that the utilization of the latter cooling schedule in SA algorithms is very effective on obtaining solutions that are either optimal or very close to the optimal. This observation has been made for values of the cooling factor α, which are between 0.95 and 0.99. For these values of α both the decrease rate is slow enough for the more efficient traversing of the solutions' surface and the convergence times achieved are faster compared to the logarithmic cooling schedule. The characteristics of the geometric cooling schedule satisfy the design objectives of this work, where speed of convergence is preferred rather than global optimality, even though high quality of clustering solutions are required.

Studying the logarithmic and geometric cooling schedules on the optimization of several of the introduced cost functions, we obtained two graphs representative of the effect of each of the cooling schedules on the optimization process. These graphs are provided below in figures 5.4 and 5.5.

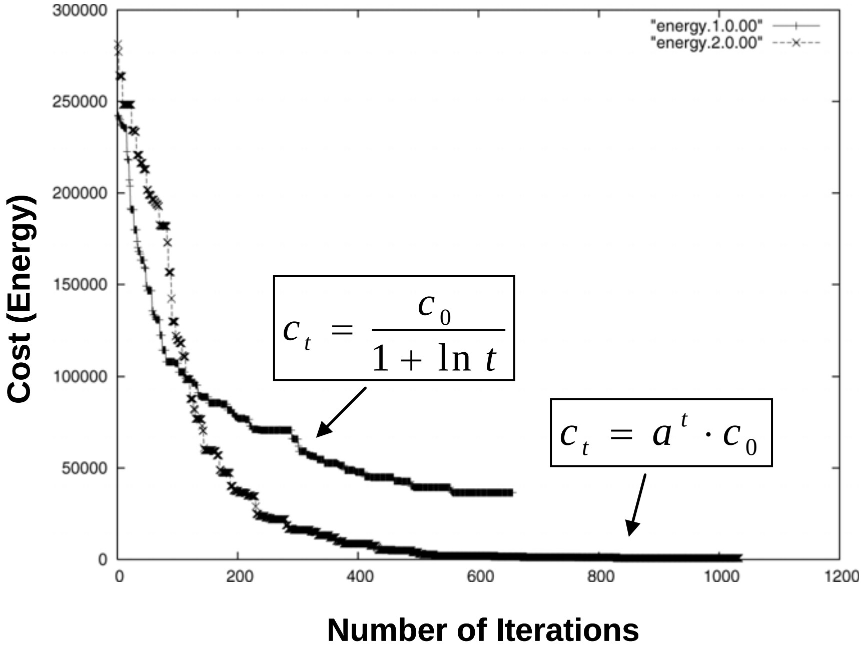


Figure 5.4. Typical relative rate of cost evolution with respect to iterations performed, by applying SA with the logarithmic and geometric cooling schedules, respectively.

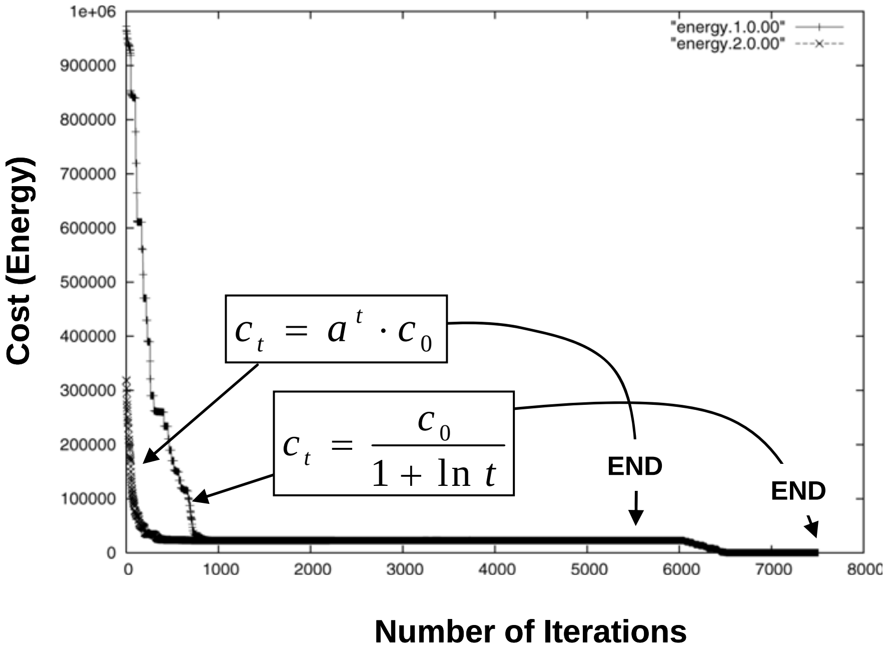


Figure 5.5. Typic relative rate of cost evolution with respect to iterations performed and optimality of solution obtained, by applying SA with the logarithmic and geometric cooling schedules, respectively.

Both figures 5.4 and 5.5 provide the typical behavior of cost evolution by applying the SA algorithm with each of the two cooling schedules on the same optimization problem. As it was expected, when SA utilizes the geometric cooling schedule the cost value improves much faster compared to the logarithmic cooling schedule. The faster rate of geometric cooling schedule does not favor the global optimality of the obtained solution compared to the logarithmic cooling schedule. This fact is highlighted from figure 5.5, where even though the progress of the optimization is faster with respect to the geometric cooling schedule, the algorithm converges to an inferior solution compared to the solution obtained from the version of SA algorithm that utilizes the logarithmic cooling schedule. Whereas, the suboptimality due to the utilization of the geometric cooling schedule does not degrade the ability of the SA algorithm to obtain solutions that satisfy the hierarchy generation objectives. Especially for the dynamic network environments under consideration the speed of convergence is more important rather than obtaining the globally optimal solution, since the quality of such a solution will not last very long due to the network dynamics.

From the experimental analysis the main conclusion is that the geometric cooling schedule satisfies the design objectives. The geometric cooling schedule based SA algorithm obtains clustering solutions fast (in combination with the termination condition), which are also satisfy the hierarchy generation objectives (sufficient optimization of the cost functions).

5.3.3 State Transition Probabilities

On every iteration the SA algorithm obtains and compares new solutions to the currently optimal one, with respect to their cost as it is evaluated from the objective function being optimized. Vital part of the SA algorithm is the generation of new solutions, which is defined from the transition mechanism. The effectiveness of the transition mechanism is important for the convergence properties of the algorithm, since it will be responsible for the speed and effectiveness of the solutions space traversing. The transition mechanism defines also the neighborhood structure of a solution.

Definition 5.4 (Neighborhood Structure): If C_i is a clustering solution and $\mathfrak{M}$ defines the transition mechanism, then the neighborhood structure V_i of the solution C_i is defined as

$$V_i = \left\{ C_j \mid C_i \xrightarrow{\mathfrak{M}} C_j \right\}.$$

V_i includes all the solutions C_j that are generated directly from the solution C_i by applying the transition mechanism $\mathfrak{M}$.

The transition mechanism involved in the design of the SA algorithm utilized in this work is based on selecting and migrating a node from one cluster to another, if this migration results in a feasible clustering solution (the clustering map consists of topological clusters, see 3.4). There are two parameters to be defined for the complete description of the corresponding transition mechanism applied. These parameters are:

- The selection method of the cluster C_i and the node $n_{i,k}$ to be migrated.
- The selection method of the cluster C_j, where the node $n_{i,k}$ will be migrated to.

Since the SA algorithm is considered a general approximation algorithm, for both of the above selections, the original implementation guidelines suggest methods that preserve the generality of the approach. In the dynamic network environments the main concern is the minimization of the time for obtaining a solution. For that reason, it is preferable to adjust the parameters of the algorithm appropriately, even though its generality properties might have to be relaxed. For the transition mechanism, the selection methods involved have been adjusted appropriately so that the algorithm converges faster without affecting the optimality of the solutions obtained. This adjustment trades off part of the generality of the algorithm for improving its speed of convergence.

In the original, generalized implementation of SA the selections for the C_i, $n_{i,k}$, C_j entities were based on the uniform distribution. Specifically, a node $n_{i,k}$ was selected randomly among the participating nodes. So, if the number of participating nodes is N the probability for selecting any of the nodes to migrate is:

$$P(n_l) = \frac{1}{N} \tag{5.3}$$

Based on these probabilities and by selecting a random number $r \sim U[0,1)$, the node $n_{i,k} \equiv n_l$ is determined as follows:

$$n_l = \left\{ l : (l-1)\frac{1}{N} \le r < l\frac{1}{N}, l \in \mathbb{Z}, 1 \le l \le N, r \sim U[0,1) \right\} \tag{5.4}$$

Along with the selection of the node $n_{i,k} \equiv n_l$, the source cluster C_i is also designated, since each node belongs to a cluster. The selection of the destination cluster C_j follows also the uniform distribution and is similar to the selection of the node to migrate n_l. If the number of generated clusters is K - included the source cluster C_i - the probability for each cluster to be selected as the destination cluster C_j is:

$$P\left(C_j\right) = \frac{1}{K-1} \tag{5.5}$$

The source cluster C_i is excluded from this selection process (e.g. there is no progress being made on the optimization process if the node is not assigned to a different cluster from its original one). With respect to the above probability and by selecting a random number $r \sim U\left[0,1\right)$ as before, the destination cluster C_j is decided as follows:

$$C_j = \left\{ j:(j-1)\frac{1}{K-1} \le r < j\frac{1}{K-1}, j \in \mathbb{Z}, 1 \le j \le K-1, r \sim U\left[0,1\right) \right\} \tag{5.6}$$

The above selection mechanisms that correspond to the original SA algorithm are independent of the cost function (hierarchy generation objectives) being optimized, in order to preserve the general nature of the algorithm. In this work, the design objectives suggest that the generality of the SA algorithm could be traded off for the improvement of the speed of convergence. Following the spirit of the latter suggestion, instead of utilizing the uniform probabilities for selecting C_i, $n_{i,k}$, C_j, customized probabilities can be applied. These probabilities could improve the convergence characteristics of the SA algorithm if they can be tailored on the cost function (hierarchy generation objectives) being optimized. The transition probabilities have to be customized appropriately so that the new solutions generated are biased towards the optimal one, with respect to the cost function optimized. By biasing the new solutions obtained and also in correlation with the termination condition the algorithm is expected to converge faster (compared to the uniform probabilities) in a clustering solution that accomplishes the hierarchy generation objectives (high quality clustering solution).

For the evaluation of the approach, one of the introduced cost functions from chapter 4 was assumed. This cost function is:

$$J(C) = \min_{C}\left(Var\left(\left|C_1\right|^2, ...,\left|C_K\right|^2\right)\right) \tag{5.6}$$

where,

Parameter	Definition
K	Number of generated clusters
C_i	Cluster *i*
$\left\|C_i\right\|$	Size of cluster *i*

The hierarchy generation objective represented from cost function (5.6) is the construction of balanced size clusters. The transition probabilities have to be adjusted to this objective by becoming aware of the cost function being optimized. For the specific cost function and hierarchy generation objectives the adjustment can be accomplished by adopting the following intuitive rules:

a) For the selection of source cluster C_i, assign higher probabilities to clusters of larger size than the optimal.

b) For the selection of destination cluster C_j, assign higher probabilities to clusters of size smaller than the optimal.

The motivation for a) as for b) is the assignment of higher probability to the migration of nodes from larger size clusters to smaller size clusters, so that the generation of balanced size clusters can be achieved faster. For the corresponding hierarchy generation objectives the probabilities assigned to the generated clusters for the selection of the source cluster C_i from where a node will be migrated depend on their sizes and are given from the following expression:

$$P\left(C_i\right) = \frac{\left|C_i\right|}{\mathrm{N}} \tag{5.7}$$

Similarly the probabilities assigned to the generated clusters for the selection of the destination cluster C_j are described as follows:

$$P\left(C_j\right) = \begin{cases} \dfrac{\mathrm{N}\text{-}\left|C_j\right|}{(\mathrm{K}\ \text{-}\ 1)\cdot \mathrm{N}\text{-}\sum\limits_{\substack{\mathrm{z=1} \\ \mathrm{z \neq i}}}^{\mathrm{K}} \left|\mathrm{C_z}\right|}, & i \neq j \\ 0 & , \ i = j \end{cases} \tag{5.8}$$

where,

Parameter	Definition
N	Number of nodes
K	Number of generated clusters
C_i	Source cluster *i*
$\left\|C_i\right\|$	Size of source cluster *i*
C_j	Destination cluster *j*
$\left\|C_j\right\|$	Size of destination cluster *j*

In each of the selections, a random number $r \sim U\left[0,1\right)$ is generated as in the original implementation. This random number along with the assigned probabilities (5.7) and (5.8) determines the source C_i and destination C_j clusters for the migration of a node and the generation of a new clustering map. Specifically, if the random number generated for the selection of the source cluster C_i is $r_i \sim U\left[0,1\right)$ then C_i is determined as follows:

$$C_i = \left\{ i : \sum_{z=1}^{i-1} P\left(C_z\right) \le r < \sum_{z=1}^{i} P\left(C_z\right), i \in \mathbb{Z}, 1 \le i \le K, r \sim U\left[0,1\right) \right\} \tag{5.9}$$

The $P\left(C_z\right)$ probabilities are given from equation (5.7).

Similarly if the random number obtained for the selection of the destination cluster C_j is $r_j \sim U\left[0,1\right)$ then C_j is specified as described below.

$$C_j = \left\{ j : \sum_{z=1}^{j-1} P\left(C_z\right) \le r < \sum_{z=1}^{j} P\left(C_z\right), j \in \mathbb{Z}, 1 \le j \le K, r \sim U\left[0,1\right) \right\} \tag{5.10}$$

The $P\left(C_z\right)$ probabilities for the selection of destination cluster are provided from equation (5.8) above.

For the generation of balanced size clusters, the customized transition probabilities determine the selection mechanism for the source and destination clusters but not for the node $n_{i,k}$ to be migrated. Since the source cluster C_i has been specified, the selection of node $n_{i,k}$ relies on the uniform distribution. Particularly, if the source cluster C_i has size $\left|C_i\right|$ then each node $n_{i,k} \in C_i$, $k \in \mathbb{Z}, 1 \le k \le \left|C_i\right|$ has the same probability to be selected for migration.

$$P\left(n_{i,k}\right) = \frac{1}{\left|C_i\right|} \tag{5.9}$$

The selection process of the node is completed by generating a random number $r \sim U\left[0,1\right)$ and applying the probabilities (5.9) to determine the node to be migrated.

$$n_{i,k}=\left\{k:(k-1)\frac{1}{|C_i|}\leq r<k\frac{1}{|C_i|},k\in\mathbb{Z},1\leq k\leq|C_i|,r\sim U[0,1)\right\} \tag{5.10}$$

The indicative speed up of SA algorithm by applying transition probabilities tailored to the cost function being optimized is represented from the following figures.

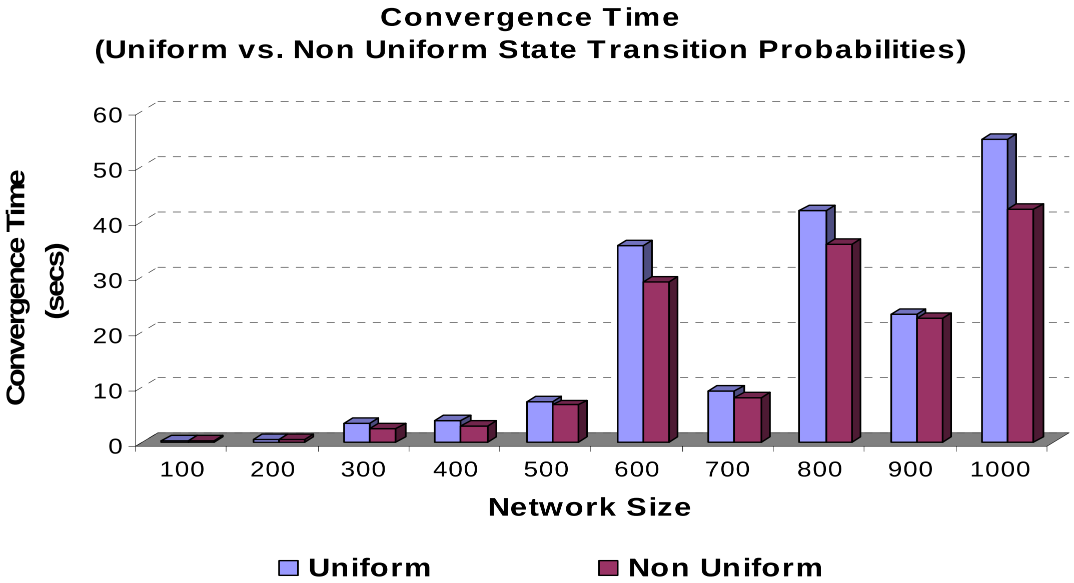


Figure 5.6. Resulting SA convergence times by applying the original (uniform) and customized (non-uniform) transition probabilities for several network sizes.

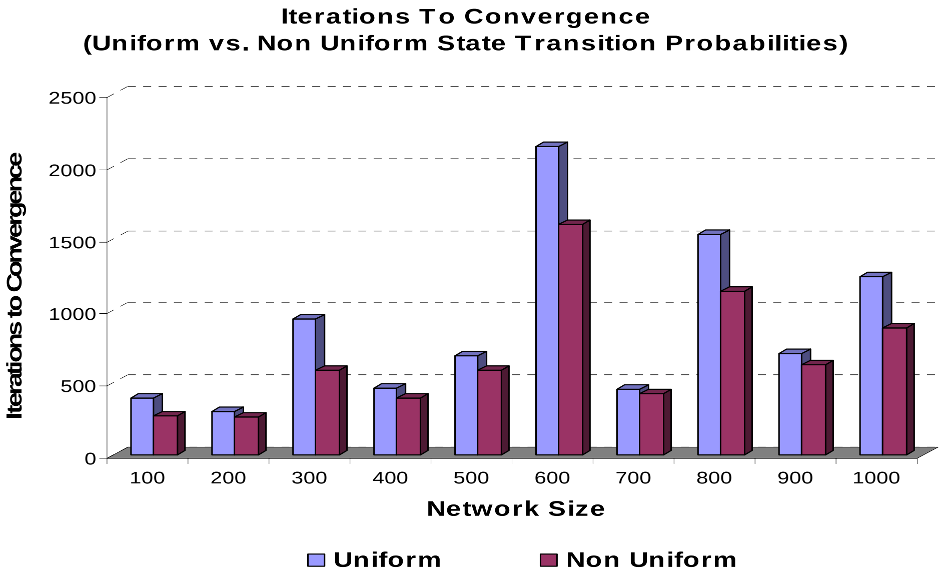


Figure 5.7. Iterations to convergence required by applying the original (uniform) and customized (non-uniform) transition probabilities for several network sizes.

Figure 5.6 presents the convergence times for several networks of various network sizes (100 – 1000 nodes) when the SA algorithm is applied with uniform and customized (non-uniform) transition probabilities. Similarly, figure 5.7 presents the required iterations until convergence for the SA algorithm when uniform and customized probabilities are being applied for the optimization of cost function (5.6) on networks of various sizes (100-1000 nodes). In both figures the results indicate the effectiveness of the approach, since the algorithm appears to converge faster when customized transition probabilities are being used for the generation of new clustering solutions. The range of the improvement is correlated with the convergence times of the original SA algorithm, which is based on the uniform transition probabilities. The larger the original convergence times, the larger is the improvement achieved. This observation favors more the utilization of the approach, since the design objectives specifically suggest the improvement of the convergence times of the algorithm when these times are prohibitive for its real time application. Furthermore, the optimization achieved from the enhanced (with the customized transition probabilities) SA algorithm, is of similar or higher quality compared to the original SA algorithm. As we have mentioned, the only drawback of the approach is that the generality of the original SA algorithm deteriorates, since the transition probabilities have to become aware of the cost function being optimized, so that they can be customized appropriately. The improvement on converge times achieved compensates for the latter compromise.

5.3.4 Generation Mechanism: Feasibility Test

The mechanism for the generation of new solutions does not only consist of the transition mechanism (transition probabilities and neighborhood structure). An equally important part of the generation mechanism is the feasibility test of the newly generated solutions. The feasibility of these solutions is defined with respect to the constraints imposed to the generated hierarchical structure. In this work the only constraint imposed to the generated clusters is to be topological (every pair of nodes belonging into the same cluster can communicate utilizing only intra-domain links). The optimization algorithm has to obtain solutions that satisfy this constraint.

Towards the convergence to the final solution, there are many clustering maps generated. Even though the final solution has to comply with the "topological clusters" constraint, the intermediate solutions obtained have to satisfy or not this constraint depending on the optimization approach. There are two classes of approaches:

- Penalty cost functions approach
- Non-penalty cost function approach

In the penalty cost functions approach, the constraint is incorporated into the cost function. A general representation of penalty cost function is provided from equation 5.11 below.

$$J_p(C) = \min_C \left(J(C) + \lambda P(C) \right) \qquad (5.11)$$

where,

Parameter	Definition
$J(C)$	Original cost function
$P(C)$	Penalty cost
λ	Constant parameter $(\lambda > 1)$

The cost of the generated clustering maps that do not satisfy the constraint (infeasible) is being penalized due to their infeasibility. Upon the optimization of the penalty cost function from SA algorithm, it is expected that the optimal solution obtained satisfies the constraint. The large cost (due to the penalty cost $P(C)$ and the amplification parameter λ) of the infeasible solutions makes them the less favorable among the clustering solutions of the solution space.

In the non-penalty cost functions approach the penalty cost part ($\lambda P(C)$) has been eliminated. The constraints– if there are any – are being imposed to the candidate solutions at each algorithmic iteration during the generation phase. The algorithm allows only the generation of feasible solutions, so a mechanism that investigates the feasibility of the newly obtained solutions is required. This mechanism functions with respect to the constraints imposed. The effectiveness of the mechanism can be evaluated from its ability to perform the feasibility test as fast and as accurate as possibly on the newly obtained solutions.

Both approaches present advantages and disadvantages. In the penalty cost functions approach, the new solutions generation mechanism is very simple and fast since it does not require any feasibility testing – the constraints have been incorporated into the cost function. On the other hand the solution space is much larger and due to the faster and lightweight version of SA algorithm, there is the risk of converging to non-feasible solutions. In the non-penalty cost functions approach, the risk of converging to an infeasible solution does not hold. All the candidate solutions provided from the generation mechanism are feasible. Compared with the penalty cost functions approach, the generation mechanism is more complicated and time consuming because along with the generation of new candidate solutions, the feasibility of these solutions has to be examined before evaluating their cost. In each iteration a new candidate solution is generated with respect to the transition probabilities and then tested for feasibility. If this solution is not feasible, it is discarded and new solutions are generated until a feasible one can be obtained, so that the SA algorithm can continue its iterations. Even though the non-penalty cost functions approach eliminates the risk of converging to an infeasible solution and the solution space appears to be much smaller than the penalty cost functions approach, the former approach appears to be more time consuming and more complicated to implement.

In this work, the non-penalty cost functions approach has been selected over the penalty cost functions one. This design choice emerged after evaluating the implementation of SA based on the latter approach. For the construction of topological clusters, a penalty $P(C)$ cost was introduced for the infeasible candidate solutions. The penalty cost is directly related to the degree of infeasibility of the candidate solutions and it is measured with respect to the number of topological partitions for each cluster.

Definition 5.5 (Topological Partition): If C_i is the i^{th} cluster, then $T_k^{C_i}$, which denotes the k^{th} topological partition (subset) of C_i, is defined from the maximal set of nodes $n_k^{C_i} \in C_i$, which are topologically connected - are connected only through intra-cluster links. If the cluster C_i consists of L topological partitions, it holds that:

- For a node $n_k^{C_i} \in T_k^{C_i}$ then $n_k^{C_i} \notin T_z^{C_i}$ for $z \neq k$.
- $\left|C_i\right| = \left|\bigcup_{l=1}^{L} T_l^{C_i}\right|$.

Definition 5.6 (Topological Clusters Penalty Cost): If C_i denotes the i^{th} cluster, $T_k^{C_i}$ is the k^{th} topological partition (subset) of C_i and $\Im_i$ is defined as the set of all subsets $T_k^{C_i}$:

$$\Im_i = \left\{ T_k^{C_i} : C_i = \bigcup_k T_k^{C_i} \right\} \tag{5.12}$$

then the penalty cost $P(C)$ is defined as:

$$P(C) = \sum_{i=1}^{K} \left(\left|\Im_i\right| - 1 \right) \tag{5.13}$$

The term $\left(\left|\Im_i\right| - 1 \right)$ suggests the existence of only one topological partition per cluster. Any number of partitions more than one is considered undesirable and contributes to the penalty cost.

As it appears from the above definition, the larger the number of topological partitions, the larger the penalty cost. Due to the type of constraint, the simplicity of generating new candidate solutions in the penalty cost function approach is compromised from the complexity of determining the topological clusters for the penalty cost evaluation. Furthermore, due to the termination condition applied, the algorithm was converging most of the times to a non-feasible solution, which is undesirable.

The infeasibility of the solutions in the penalty cost functions approach was crucial for adopting the non-penalty cost functions approach. Despite the more complicated generation of new candidate solutions mechanism, the algorithm guarantees the convergence to feasible solutions. Furthermore, the complexity of the generation mechanism in the non-penalty cost functions approach has been shifted to the evaluation of the cost in the penalty cost functions approach. So, the latter approach does have any advantages over the former one.

By adopting the non-penalty cost functions approach, the generation mechanism had to be implemented efficiently in order to have the minimal effect possible on the convergence time of the algorithm. As it appeared to be from the evaluation of the resulted convergence times, the feasibility test of the generation mechanism is the dominant part of the SA algorithm for the specific type of constraint imposed. Due to the dominating effect of the feasibility test, the convergence time of the algorithm was varying significantly among several implementation of the mechanism. The following graphs, which present the convergence times of the SA algorithm with respect to network size and number of clusters generated, are indicative of the latter observation.

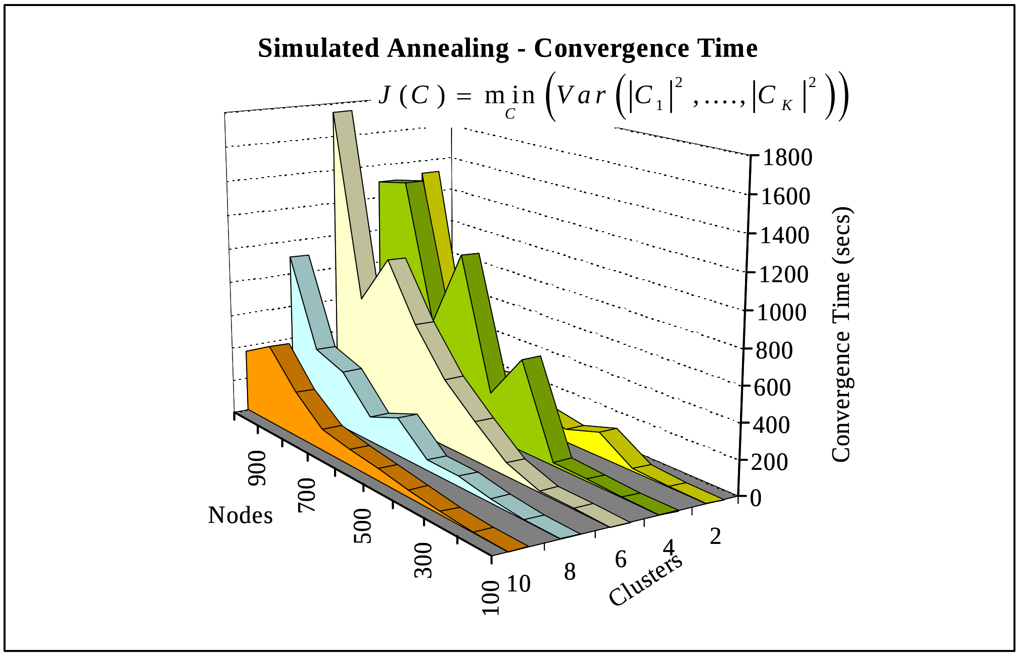


Figure 5.8. Convergence time of SA algorithm with respect to network size and number of clusters generated when inefficient feasibility test mechanism is applied.

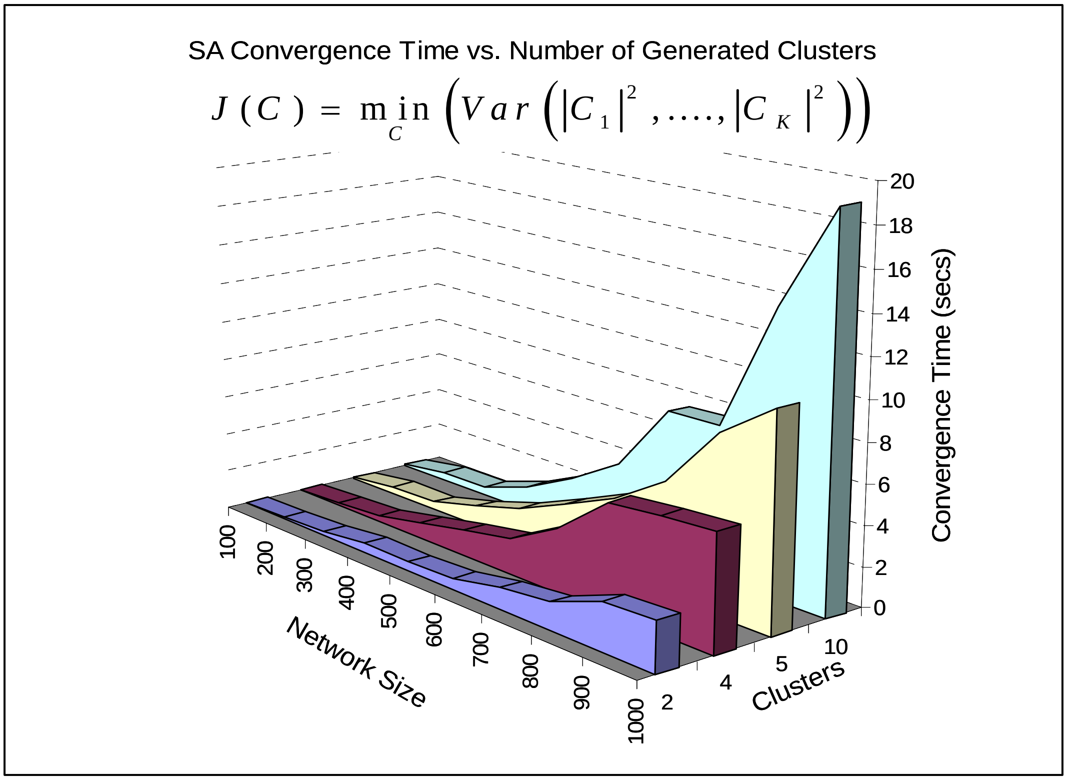


Figure 5.9. Convergence time of SA algorithm with respect to network size and number of clusters generated when efficient feasibility test mechanism is applied.

```
for every cluster $C_i$
    begin
        Initialize the sets
        $V$ , which contains the nodes
        $n_k^{C_i} \in C_i$ not assigned to any topological partition $T_z^{C_i}$ of cluster $C_i$
        and
        $V' = \{\varnothing\}$ , which contains the nodes $n_k^{C_i} \in C_i$ assigned to any of the
        topological partitions $T_z^{C_i}$ of cluster $C_i$
        **Initialization**
        $V = \{n_k^{C_i} : n_k^{C_i} \in C_i, 1 \le k \le |C_i|\}$
        $V' = \varnothing$
         **Main loop**
         Insert a node $n_j^{C_i} \in V$ to $V'$
         $V = \{n_k^{C_i} : n_k^{C_i} \in C_i, 1 \le k \le |C_i|, k \ne j\}$
         $V' = \{n_j^{C_i}\}$

          for every node $n_z^{C_i} \in V'$
             begin
                Insert in $V'$ any of the one hop neighbors $n_l^{C_i}$ of $n_z^{C_i}$ , for
which
                holds that: $n_l^{C_i} \in V$ ( $n_l^{C_i} \notin V'$ )
             end

        if $V = \varnothing$ and $|V'| = |C_i|$
             $C_i$ is feasible
        else
             $C_i$ is infeasible
      end

if every cluster $C_i$ is feasible
     the candidate clustering  solution generated is feasible
```

Figure 5.10. Pseudo code implementing the efficient feasibility test mechanism

Figure 5.8 represents the convergence times The only difference between the two versions of SA algorithm is the implementation of the feasibility test mechanism. As figure 5.8 shows, the implementation that corresponds to penalty cost functions approach is the least efficient one and is based on processing of *lists* for determining the topological partitions for each cluster. The implementation that corresponds to figure 5.9 is the most efficient one and corresponds to *lookup* processing. Even though the *lookup* processing is based on *arrays*, which requires more memory compared to *lists*, it is much more effective on improving the convergence time performance of SA algorithm. Specifically, the convergence times between the two implementations differ significantly. The implementation that corresponds to figure 5.8 and is based on *lists* processing requires almost 30 minutes to complete for networks of 1000 nodes, as opposed to the less than 20 seconds convergence time required from the implementation that corresponds to figure 5.9 and is based on *lookup* methods. The pseudo code that implements the efficient feasibility test that corresponds to figure 5.9 is provided in figure 5.10.

5.3.5 Initial Solution

The main design objective for the hierarchy generation framework is the real time convergence to clustering solutions that satisfy the pre-specified hierarchy generation objectives. In this section one more adjustment on the functionality of SA algorithm is proposed. This adjustment is related to the initial clustering solution utilized for bootstrapping the algorithm. In the original implementation of SA algorithm, the only requirement is the optimality of the solution obtained (real time convergence is not considered), the algorithm is bootstrapped with a randomly obtained initial solution. For the clustering problem at hand, the only requirement imposed to the randomly obtained initial solution is its feasibility with respect to the constraint of topological clusters. The adjustment proposed is to bootstrap the algorithm with an initial solution which has better cost than a randomly obtained one with respect to the cost function being optimized. Even though, if we apply such an initial clustering solution the improvement on the convergence time of SA algorithm seems intuitive, it is not. Due to its randomization nature, the SA algorithm searches randomly the surface of solutions and at some extent accepts deteriorations in cost instead of only improvements. Hence, even if the SA is bootstrapped with a better initial solution, it might converge slower. For this reason, prior to adopting the "better than random initial solution" adjustment, the effectiveness of the approach has to be investigated. There are two issues to be addressed:

1. The level of the improvement we get with respect to the quality of the initial solution.
2. How we can generate initial solutions that will provide us with large convergence time improvements.

In this work the firs issue will mainly addressed in detail, since it is important for determining the effectiveness of the approach and is incorporation for enhancing the SA algorithm. As we have mentioned, due to the randomization character of SA, it is not straightforward that by starting from a better than a random initial solution will result to any convergence time improvement at all. The latter has been investigated by quantifying the convergence time effect with respect to the quality of the initial solution. Specifically, SA has been bootstrapped with a better than random initial clustering solutions and its convergence time has been determined. From the collection of these results has been observed that despite the randomization character of the algorithm, by starting from a better than a randomly selected clustering solution (i.e. with respect to the cost function being optimized), the convergence time is improved. For quantifying the improvement on the convergence time, some indicative results with respect to various qualities of initial solutions are provided below. These results were collected for networks of 100 and 200 nodes. Samples of such networks are being demonstrated in figure 5.11.

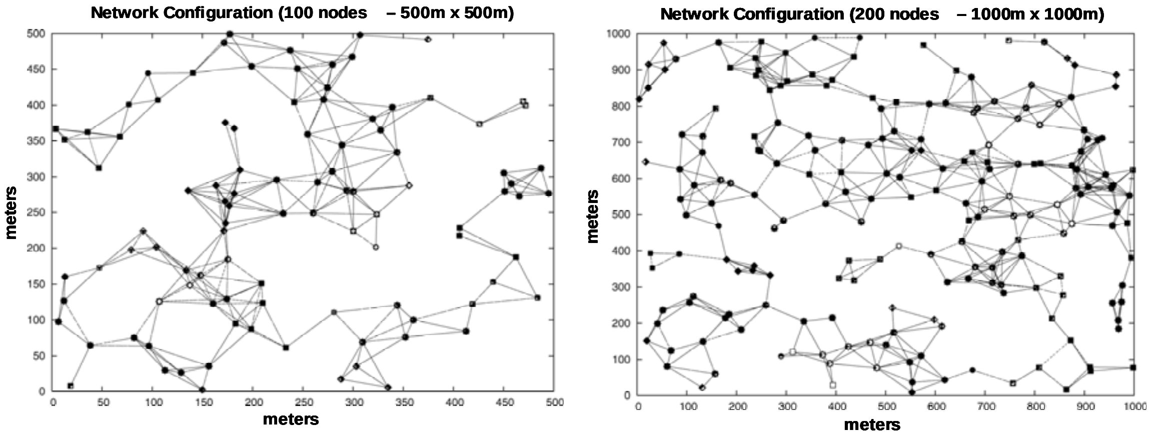

Network Configuration (100 nodes – 500m x 500m)
meters
meters
0
50
100
150
200
250
300
350
400
450
500
Network Configuration (200 nodes – 1000m x 1000m)
meters
meters
0
100
200
300
400
500
600
700
800
900
1000

Figure 5.11. Sample networks of size 100 and 200 nodes

The methodology applied for the collection of results is based on the networks presented above and the SA clustering algorithm implementation described from the block diagram of figure 5.1. The cost function (equation 5.6) for the generation of balanced size clusters was utilized. The cost of a random generated solution with respect to this cost function was computed and then sample clustering solutions with cost that was fraction of the cost of this random solution were generated.

The y-axis of the following figures represents the convergence time in seconds for various qualities (costs) of initial solutions, and they are characterized from the fraction of their cost compared to the random initial solution. The x-axis is marked with the value of this fraction. The convergence times represented from the following graphs are indicative for the cost function (equation 5.6) but they provide some very useful observations for the proposed adjustment. The following results have been averaged out after a large number of runs $O(1000)$.

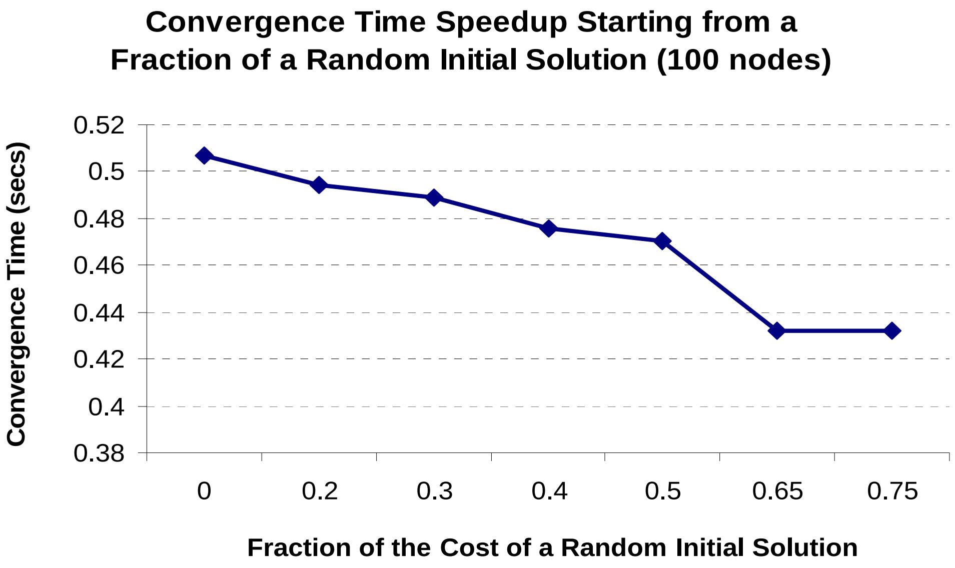


Figure 5.12. SA convergence time improvement with the quality of the initial solution for 100 nodes network

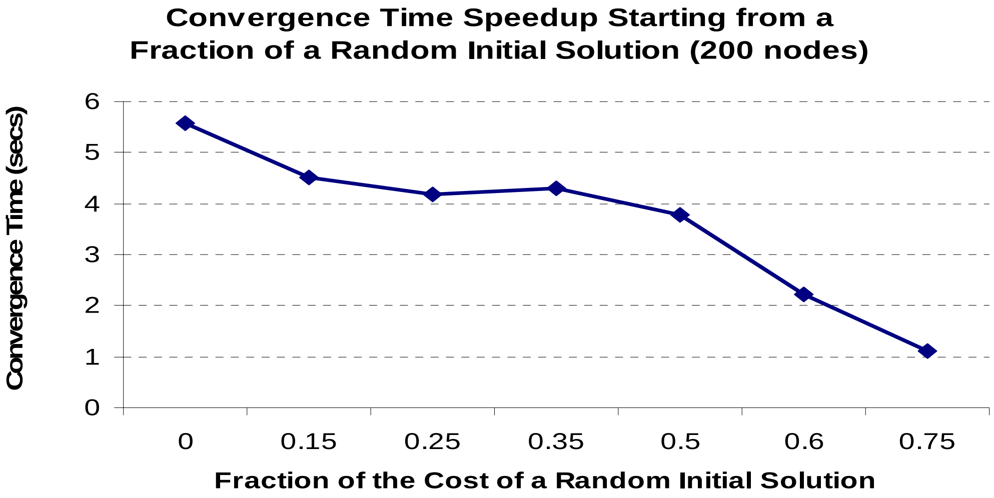


Figure 5.13. SA convergence time improvement with the quality of the initial solution for 200 nodes network

The most important conclusion can be drawn from the above figures is that the bootstrapping of SA algorithm with a better than a random initial solution has advantageous effect on the convergence time of the algorithm. Both curves in figures 5.12 and 5.13 respectively present a dropping tendency with the improvement of the cost of the initial solution compared to the cost of the random one. Furthermore, important conclusions can be drawn also from the quantification of the convergence time improvement. By comparing the results of figures 5.12 and 5.13 respectively, there is an indication that the larger the network, the larger appears to be the improvement on the convergence time. For the sample network of 100 nodes utilized in this simulation analysis, the convergence time drops ~100ms when the clustering solution has cost 75% better compared to the random one. Whereas, for the sample network of 200 nodes the convergence time improves by ~4.8s when the initial clustering solution presents 75% better cost compared to the random one. Furthermore, to quantify better this decrease (improvement) on convergence time, the percentage of improvement is provided in the following table.

Network size	**Actual Time Improvement**	**Percentage of Improvement**
100	0.1s	19.2%
200	4.8s	82.8%

Table 5.1. Percentage improvements on convergence time

By looking at the corresponding percentages of improvement on convergence time when SA starts from a "*better than random initial solution*", it is obvious that the larger the network, the larger the improvement. The latter can be explained from the much slower convergence times presented from the original SA algorithm when the network size increases, so there is more space for improvement. The proposed adjustment is sufficient to improve enough the convergence time and constitute a traditionally slow approximation algorithm, realizable and applicable in dynamic environments like the MANETs.

Since there are indications (e.g. based on the simulation analysis results), that despite the randomized search of SA algorithm's towards the optimal solution, the convergence time is improved by starting from a good initial solution, then mechanisms that will generate the appropriate initial solutions have to be suggested. This work does not explore this problem in depth, since the applied mechanisms must be aware of the hierarchy generation objectives (cost function being optimized). Initial solutions that can improve the convergence time can be generated from heuristic methods customized to the hierarchy generation objectives (i.e., for the generation of balanced size clusters we can generate initial solutions utilizing a customized min-cut algorithm). Also, modified optimization algorithms can be useful for the generation of quality initial solutions. Furthermore, a feasible, previously generated optimal solution from SA can be utilized for bootstrapping. Due to the dynamics of MANETs environment, the clustering decisions have to undergo corrections in order to retain their optimality with respect to the topology changes. In the case where the SA have to be reapplied, then instead of generating a new initial solution a previously optimal one can be used for bootstrapping, under the condition that it is still feasible with respect to the new topology. The latter approach can provide quality initial solutions especially when the topology is slowly changing with respect to the reapplication frequency of SA algorithm.

An important consideration that has to be made for the selection of an efficient initial solution generation mechanism is that the combined time for the generation of the initial solution and the convergence of SA has to be smaller than the convergence time of SA when it is bootstrapped with a randomly selected initial solution. Hence, the following inequality has to be satisfied at all times in order not to eliminate the advantageous effect of initial solution on the convergence time of the algorithm.

$$\begin{gathered} T_{nris} \leq T_{ris} \Rightarrow t_{gnris} + t_{SA} \leq t_{gris} + t_{SA}^{'} \Rightarrow t_{gnris} - t_{gris} \leq t_{SA}^{'} - t_{SA} \Rightarrow \\ \Rightarrow \Delta t_{gis} \leq \Delta t_{SA} \end{gathered} \tag{5.14}$$

where,

Parameters	Description
$nris$:	Non-random initial solution
ris :	Random initial solution
T :	Complete process time
t :	Partial process time

5.3.6 Energy Updates

The basic functionality of SA algorithm is based on the cost evaluation of the candidate clustering solution $C(t)$ obtained from a generation mechanism in every algorithmic iteration $t\left(t \in \mathbb{Z}^{+}\right)$. The cost of the candidate solution is compared with the cost of the currently optimal one C_t^* and depending on their difference

$$\Delta E = J\left(C_t^*\right) - J\left(C(t)\right) \tag{5.15}$$

the Metropolis criterion

$$P_{c_t}\left(C_{t+1}^{*} \leftarrow C(t)\right)=\begin{cases}1 & \text{if } \Delta E>0 \\ \exp\left(\dfrac{\Delta E}{c_t}\right) & \text{if } \Delta E \leq 0\end{cases} \tag{5.16}$$

decides on which of the solutions ($C(t)$ or C_t^*) will prevail (e.g. will be carried on as the optimal C_{t+1}^* in the $(t+1)^{th}$ iteration). Obviously, every time a new candidate solution is generated, its cost must be evaluated. This evaluation requires computation time, so inevitably contributes to the convergence time of the SA algorithm. Intuitively, the contribution becomes more significant for larger optimization problems (i.e. large network sizes). This is because more iterations are required for the convergence of the algorithm, so more candidate solutions are being generated whose cost has to be evaluated.

Since the design objectives suggest the improvement of the speed of convergence of the algorithm, a possible adjustment that could reduce the convergence time is the efficient evaluation of the candidate solutions cost. Based on the generation mechanism principle, where in the t^{th} iteration a new candidate solution $C(t)$ is obtained by perturbing the currently optimal one C_t^*. The perturbation is characterized from the transition probabilities and the neighborhood structure mentioned above. The cost of the currently optimal solution C_t^* is known and is $J\left(C_t^*\right)$. In order to compute the cost $J\left(C(t)\right)$ of the new candidate solution $C(t)$, only the contribution (update) of the perturbation on the currently optimal cost $J\left(C_t^*\right)$ have to be specified. Specifically, the mechanism that is proposed is instead of determining the cost of the candidate solution generated at each iteration, is to compute the update to the cost, that results from the perturbation. This mechanism can be incorporated into the functionality of SA algorithm as follows:

Initial Iteration: Generate an initial clustering solution C and evaluate its cost $J\left(C\right)$ with respect to the cost function being optimized. Since this is the initial iteration, SA marks this solution as the currently optimal one C_0^* and its cost $J\left(C_0^*\right)$ is the currently optimal one $J\left(C^*\right)$.

Follow Up Iterations: Assume that on the $t^{th} (t > 0)$ iteration, the optimal clustering solution is C_t^* and its corresponding cost is $J\left(C_t^*\right)$. By perturbing C_t^* we obtain a new candidate solution $C\left(t\right)$. The corresponding cost of this solution $J\left(C\left(t\right)\right)$ can be computed utilizing the contribution ΔE_p of the perturbation p on the cost of the currently optimal clustering solution:

$$J\left(C\left(k\right)\right) = J\left(C_k^*\right) + \Delta E_p \tag{5.17}$$

So, instead of having to compute $J\left(C\left(k\right)\right)$ from scratch, we just have to compute the difference ΔE_p of the cost due to the perturbation p.

Due to this mechanism, the only time that the entire clustering solution has to be taken into consideration for the cost computation $J\left(C\right)$ is during the initial iteration. Afterwards the computation is done based on the cost differences (updates) ΔE_p.

For the evaluation of the efficiency of the adjustment on the convergence time of the algorithm, two cost functions for the generation of balanced size clusters have been utilized. Their description follows:

$$J(C) = \min_{C}\left(Var\left(\left|C_1\right|^2, ..., \left|C_K\right|^2\right)\right) \quad (5.18)$$

and

$$J(C) = \min_{C}\sum_{i=1}^{K}\left|C_i\right|^2 \quad (5.19)$$

where,

Parameter	Definition
K	Number of generated clusters
C_i	Cluster *i*
$\left\|C_i\right\|$	Size of cluster *i*

For each of the cost function the update function ΔE_p has to be defined, so that it can be utilized for the computation of the new candidate solutions cost. This function depends on the perturbation method applied for the generation of the new candidate solutions in each SA iteration. As it was mentioned, the basic principle of the perturbation mechanism is the migration of a member node from a cluster $C_i^*(t)$ to another cluster $C_j^*(t)$ of the currently (t^{th} iteration) optimal clustering solution $C^*(t)$, subject to the constraint that the new candidate solution must be topologically feasible. Thus, with respect to this perturbation mechanism and the cost functions (equation 5.17 and equation 5.18) being optimized, the update function ΔE_p is defined appropriately:

For equation 5.17

$$\Delta E_p(t) = \frac{1}{K-1}\left(\Delta E_1(t) - \Delta E_2(t)\right) \quad (5.20)$$

where

K: number of generated clusters

$$\Delta E_1(t) = \left(\left|C_i^*(t)\right| - 1\right)^4 + \left(\left|C_j^*(t)\right| + 1\right)^4 - \left|C_i^*(t)\right|^4 - \left|C_j^*(t)\right|^4 \quad (5.21)$$

$$\Delta E_2(t) = 4\alpha\left(\overline{\left|C^*(t)\right|^2} + \frac{\alpha}{K}\right) \quad (5.22)$$

$$\alpha = \left|C_j^*(t)\right| - \left|C_i^*(t)\right| + 1 \quad (5.23)$$

$$\overline{\left|C^*(t)\right|^2} = \frac{1}{K}\sum_{i=1}^{K}\left|C_i^*(t)\right|^2 \quad (5.24)$$

For equation 5.18

$$\Delta E_p(t) = -2\left(\left|C_i^*(t)\right| - \left|C_j^*(t)\right| - 1\right) \quad (5.25)$$

Using the relations above we evaluate ΔE_p which determines through the Metropolis criterion the optimal clustering solution of the $(t+1)^{th}$ iteration.

For the evaluation of the cost updates method the above expressions were applied for the cost computation in every iteration of SA algorithm. For both cost functions, the average SA convergence times for several networks were determined. These values

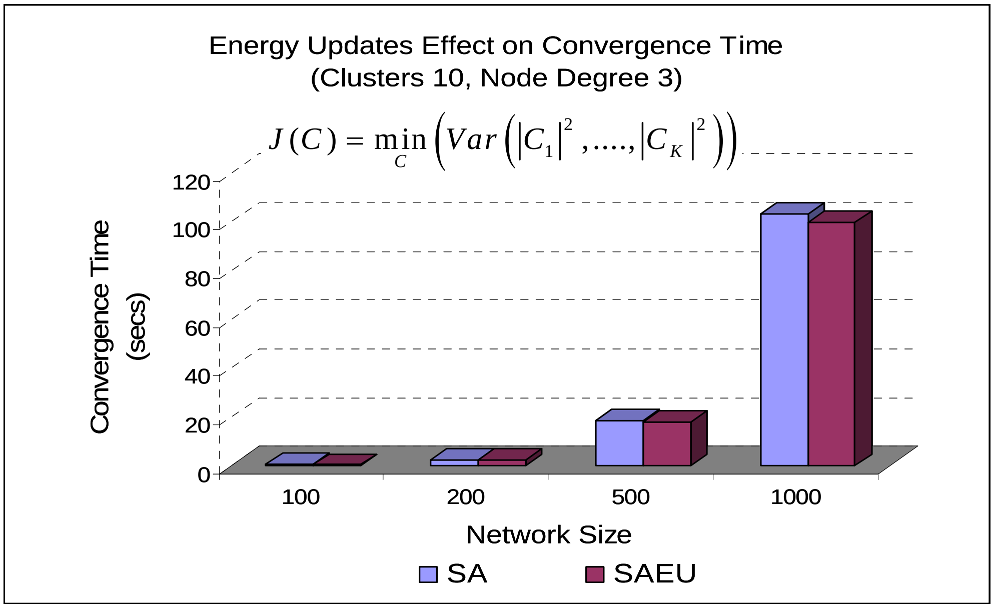


Figure 5.14. Expected Convergence Times Comparison for SA and SA with Energy Updates (SAEU) for the generation of balanced size clusters (cost function 5.18)

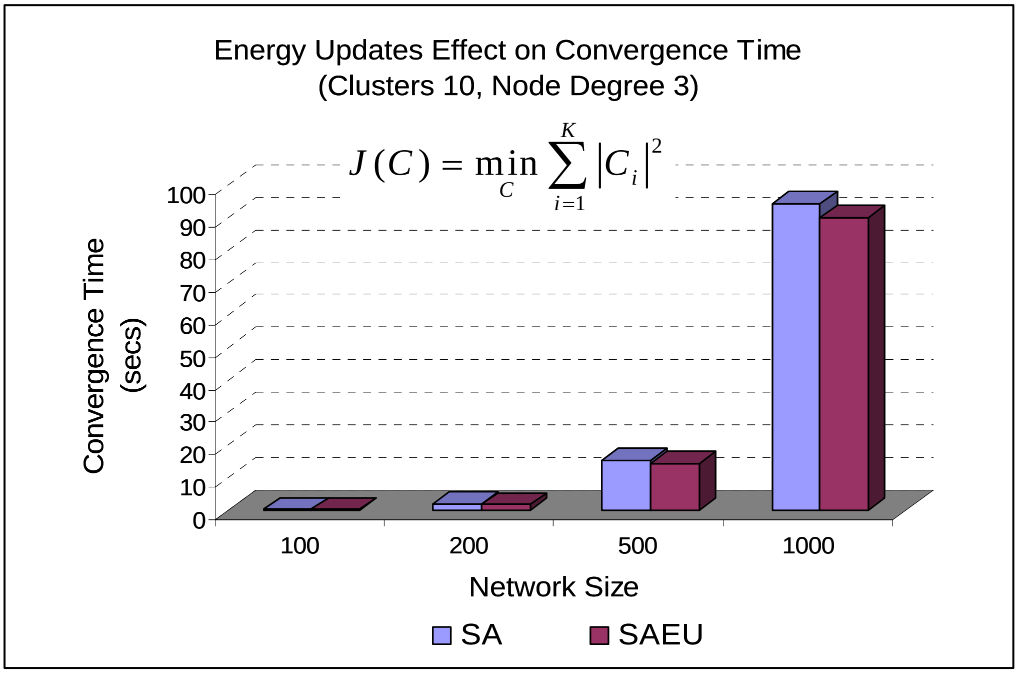


Figure 5.15. Expected Convergence Times Comparison for SA and SA with Energy Updates (SAEU) for the generation of balanced size clusters (cost function 5.19)

were compared with the average convergence times for the same set of networks when the method of cost updates is not applied (e.g. original implementation of SA algorithm).

The results above indicate small improvement on the convergence time of SA algorithm, when the energy updates methods is utilized. Specifically, for scenarios (network sizes 100 and 200 nodes) where the convergence time of the algorithm is already fast, the improvement is negligible. For scenarios (network sizes 500 and 1000 nodes), where the convergence time is larger, there is a noticeable improvement but still it is not significant. This behavior can be explained from the dominating contribution of new candidate solutions generation phase on the convergence time of the algorithm– as it was explained in a previous subsection. The contribution of the cost computation on convergence time is not significant compared to the generation mechanism’s contribution. Due to this, any improvement on the cost computation mechanism is not expected to contribute in significant changes (improvements) on the convergence time of the algorithm. Moreover, offline processing is required in order to obtain the update function $\Delta E_p\left(t\right)$ for every cost function being optimized. In conclusion, the method reduces the generalized character of SA algorithm for insignificant improvements on the convergence time. On the other hand, there are scenarios (large networks - large solution spaces – large convergence times) where even small improvements are important and beneficial for the real time applicability of the algorithm.

5.4 Convergence Times of the Adjusted SA Algorithm

Several adjustments on the parameters and on the functionality of the original SA algorithm have been proposed for reducing the time required to obtain solutions that satisfy a set of pre-specified hierarchy generation objectives. During the presentation of these adjustments, the effect on the convergence time of each one of these separately has been evaluated. In this section comprehensive results for the convergence time of the SA algorithm are provided. These results correspond to the implementation of SA that has been adjusted with respect to the collection of adjustments suggested and are indicative of the convergence time performance of the algorithm. The first of the following two graphs represents the convergence time of the algorithm with respect to network size and number of clusters being generated. The second graph represents the convergence time of the algorithm with respect to the network size and the average node degree.

Specifically, figure 5.16 represents the convergence time of the adjusted SA algorithm for several network sizes, varying from 100 to 1000 nodes and different numbers of generated clusters, which vary from 2 to 10 clusters. All the networks utilized for the collection of the following results present average node degree (average number of participating nodes' one hop neighbors) equal to 10.

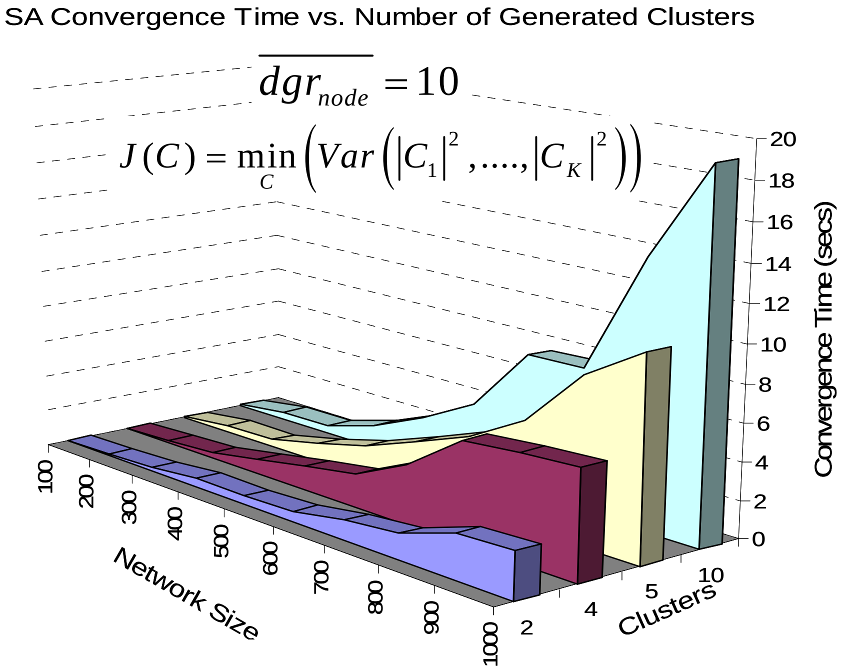


Figure 5.16. Convergence times of the adjusted SA algorithm with respect to various network sizes and number of generated clusters (average node degree equals to 10)

From figure 5.16 interesting observations can be drawn. The most important one is the ability of the algorithm, for large number of scenarios, to converge very fast to efficient clustering solutions (e.g. few *msecs* to few *secs* – for 200 nodes less than 1 second is required). These scenarios include networks sizes of few hundreds of nodes. Equally important on the convergence time of the algorithm is the number of generated clusters. The larger the number of generated clusters, the larger the convergence time of the algorithm for the same network size. In conclusion the adjusted SA algorithm can be applied successfully and efficiently in real time for a majority of scenarios (combination of network size and clusters generated) even for highly dynamic networks. For other scenarios, with larger solution spaces, the algorithm presents larger convergence times, which are still not prohibitive for networks with slower dynamics (the topology changes do not happen very frequently).

As the network size and the number of generated clusters, the average node degree of the network is equally important for the convergence time of the algorithm. This is indicated from the results pictured on the following figure 5.17.

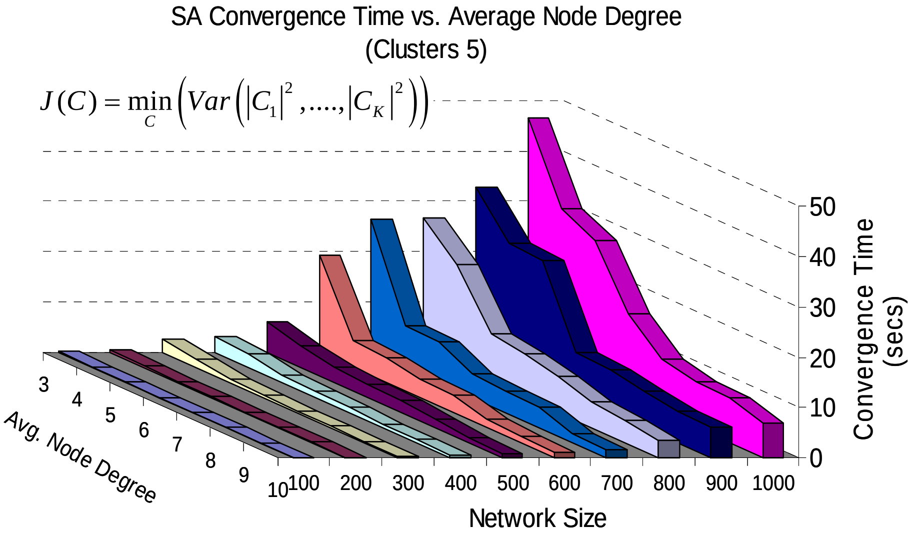


Figure 5.17. Convergence times of the adjusted SA algorithm with respect to various network sizes and average node degrees (the number of generated clusters equals to5).

The main observation is that the larger the average node degree, the faster is the convergence time of the algorithm for the same network size and number of generated clusters. This result can be justified due to the topological constraints imposed and the generation of new candidate clustering solutions mechanism. The latter mechanism releases only feasible solutions to be compared against the currently optimal solution at every algorithmic iteration. For small average node degree the generation mechanism runs into the risk of becoming very slow due to the sparse nature of the network and the inherent difficulty to locate fast new feasible candidate solution. This observation along with the dominant effect of the mechanism on the convergence time of the algorithm result on the effect pictured on figure 5.17. In contrary, new feasible solutions are obtained faster in dense networks (large average node degree), so the algorithm is expected to converge faster (e.g. for average node degree $\overline{dgr} = 10$ the algorithm requires less than 20 *secs* to generate 10 clusters ina network of 1000 nodes).

The adjusted SA algorithm appears to be efficient, scalable and applicable in real time for a majority of scenarios. Furthermore, the parameters (network size, average node degree and number of clusters to be generated) of the network under consideration are important for the speed of convergence of the algorithm. The latter observation may lead to suggestions for modifying dynamically the network, so that the targeted convergence time is achieved and the network dynamics are captured. If such an approach could be adopted, the results of figures 5.16 and 5.17 could be utilized for specifying the network parameters, which achieve better convergence times.

Chapter 6: Metrics Based Distributed Domain Generation Algorithm

6.1 Introduction

The Simulated Annealing based domain generation framework presented in the previous chapters has been adjusted so as to provide rapidly clustering solutions that satisfy the instructed hierarchy generation objectives. On the other hand, due to its centralized nature, the SA based clustering framework will not be able to perform adequate in highly dynamic environments. Since the study of such environments is within the scope of this work, efficient hierarchy generation mechanisms must be provided.

For efficiently capturing the dynamics of fast changing networks, distributed hierarchy generation algorithms must be designed. Their ability to generate hierarchical structures based only on the collection and exchange of local information makes them favorable for highly dynamic environments. On the other hand, the hierarchical structures obtained, are not expected to have the quality (cost) of the structures provided from the SA-based mechanism, due to the localized hierarchy generation decisions. In such rapidly changing environments the speed of the algorithm is more important that the quality of the solution, since the hierarchical structures generated are expected to be short-lived because of the frequent changes on the topological map of the network. But still, it is crucial for the hierarchical structures obtained to satisfy the set of pre-specified hierarchy generation objectives, so that the hierarchy is beneficial to the performance of the network and not harmful.

For dealing with highly mobile environments and aiming on accomplishing the hierarchy generation objectives, a distributed hierarchy generation algorithm has been designed, in accordance to the spirit of SA-based algorithm. The functionality of the designed algorithm is based on the exchange of one hop information. This information is related to the hierarchy generation objectives and is expressed via the utilization of the appropriate metrics presented in chapter 4. For the presentation of the distributed algorithm specific hierarchy generation objectives and the corresponding metrics have been selected. Specifically, the generation of similar to mobility domains is imposed and the utilization of the mobility metrics is required. Due to the enforcement of the similar mobility objective and the utilization of mobility metrics, the corresponding version of the distributed hierarchy generation algorithm is being referred as “mobility based distributed generation algorithm (DGA)”

The main objective of this algorithm is the generation of stable hierarchical structures by grouping together the nodes that present similar mobility characteristics. Doing so, it is expected that the nodes of the same group will remain connected for long periods of time, reducing significantly the membership changes and the resulted maintenance overhead. If the maintenance overhead is reduced, the performance of the network will improve, benefiting from the hierarchical application of the various networking protocols onto a stable hierarchical structure.

The algorithm is based on one-hop information exchange and presents *O(n)* communication complexity in a network of *n* nodes. The generated clusters appear to be more robust compared to some well known existing distributed clustering algorithms. Furthermore, the algorithm presents very promising performance characteristics in cases of large and highly mobile networks, where the existing distributed algorithms fail. The ability of the algorithm to handle the dynamics of the network emerges from its inherent functionality to group the nodes with respect to these dynamics.

6.2 Overview of the mobility based DGA

In this section the principal operation of the proposed mobility based domain generation algorithm is presented. An example is also given, which demonstrates the algorithmic steps followed from DGA for generating a robust to mobility hierarchical structure.

6.2.1 Mobility Based Distributed Generation Algorithm (DGA)

The mobility based DGA is based on one-hop information exchange. The information is related to the mobility metrics we introduced in section 3. We assume that each node represented from a unique ID, can obtain information about its speed, direction and position (e.g., only in the case where the metric of interest is the Link Expiration Time (*LET*)). Also, the set of their one-hop neighbors can be obtained from the exchange of link state information or the transmission of heartbeat messages.

In the proposed algorithm each node joins a domain after having gone through three phases. The objective suggests that the generated domains consist of similar nodes with respect to their mobility characteristics. The three phases of the algorithm towards the generation of hierarchical structure are:

- **Phase I – Neighbor Selection**

In this phase each node broadcasts its ID and information (direction, speed, position) related on the decision making metric (relative direction, relative velocity, LET) to its one-hop neighbors. Each node, after the collection of the appropriate information is able to determine the value of the metrics (mobility in this case) of interest for each one of its neighbors. For each node, the set of these values determines the one hop neighbor that the corresponding node will select to join for the formation of a domain. Specifically for the mobility based DGA, where the metrics of interest could be the relative direction or the relative velocity, a node will select the one hop neighbor that corresponds to the lowest metric value – in the case of LET the neighbor that corresponds to the highest value is the dominant candidate. In the case where multiple dominant candidates are present (the same lowest value corresponds to more than one hop neighbors), a tiebreaker rule is used (e.g. ID of the neighbors or random selection) for resolving the conflict, since only one neighbor has to be selected for the formation of a domain.

- **Phase II – *InfoExchange* List Composition**

After having selected the most appropriate neighbor to form a domain with, each node informs the selected neighbor node for this decision. Each node collects the decisions related to him and records the IDs of the neighbor nodes who have selected him. After the collection of these messages, each of the nodes generates the *InfoExchange* list, which is the union of the node IDs collected in Phase II, the ID of the neighbor he has selected and his own ID. The *InfoExchange* list is sorted in ascending order. The ordering of the *InfoExchange* list is very important for the convergence of the algorithm since it provides the basis for the distributed synchronization of the participating nodes.

- **Phase III – Domain Formation**

For the domain formation, each node does not have to communicate with every neighbor but only with those in the *Infoexchange* list. In the distributed environment the sorted *Infoexchange* list is utilized for synchronization among the nodes. A node has to wait for the nodes with lower ID in the *Infoexchange* list to decide on the domain to join and then has to communicate its selection. If the ID of the node is the lowest in the list, then forms a cluster with this ID and communicates it to its neighbors that exist in the list. By the completion of this phase each node belongs to a domain characterized by a unique domain ID – these IDs corresponds to node IDs which have been assumed unique.

After the high level overview of the various phases, the pseudo-algorithm provided below reveals the detailed functionality of the proposed algorithm and of its various phases as they are performed from each one of the participating nodes:

Phase I

Step I: (Communication)

Broadcast to 1-hop neighbors a TYPE I message of the form:

(*myID*, *Information*)

The information values can be the speed, direction or position of the node and depend on the metric of interest (e.g. relative direction, relative speed, LET)

Step II: (Processing)

Collect the (TYPE I) messages from 1-hop neighbors.
Based on the information collected, evaluate the metric of interest for each one of the neighbors. With respect to the metric values obtained determine which one of the neighbors is the most appropriate for grouping with in the cluster formation.

Phase II

Step III: (Communication)

Broadcast to the selected neighbor (e.g. neighbor with the best metric value) a TYPE II message of the form:

(*myID*, *neighborID*)

Step IV: (Processing)

Collect all TYPE II messages that are referred to my ID.
Generate the *SelectedFromList* list which contains the *neighborIDs* that have selected *myID* as the preferred neighbor to be clustered with. Then generate the *InfoExchangeList*:

$$\begin{aligned}&\mathit{InfoExchangeList} =\\&\mathit{SelectedFromList} \cup \mathit{neighborID} \cup \mathit{myID}\end{aligned}$$

Sort in ascending order the *InfoExchangeList* with respect to the node IDs.

Phase III

Step V: (Communication)

```
if myID = Head(InfoExchangeList) then
  {  myCID = CID
     Send to every node with
     $nodeID \in InfoExchangeList$
     a TYPE III message of the form:
                        (myID, CID=myID)
  }
else{

      Until the reception of a TYPE III message from the nodes with:
$nodeID \in InfoExchangeList \wedge nodeID < myID$
      {  Upon the reception of TYPE III message {
            if $myCID = \varnothing$ then


              myCID = CID
            else
              if myCID > CID then
              myCID=CID
            }
      My turn to transmit:
      Send myCID to every node with
         $nodeID \in InfoExchangeList$

  Until the reception of TYPE III message from all the
  nodes with:
              $nodeID \in InfoExchangeList \wedge nodeID > myID$
  {
     Upon the reception of a TYPE III message {
      if myCID > CID {
           myCID=CID
         send to all nodes with
           $nodeID \in InfoExchangeList$
                  a message revealing my cluster selection
                        (myID, myCID)
      }}}}
```

6.3 Mobility Based DGA: Example

In this section we complete the description of the mobility based DGA by providing an example to demonstrate the domain generation algorithm and its various phases for the construction of hierarchical structure, which is robust to mobility by grouping together nodes with similar mobility characteristics. Assume that we have the network of figure 6.1, consisting of 7 nodes. Assume also that there are two groups of nodes, which consist of nodes with similar mobility characteristics. The one group consists of the nodes 1, 2, 4 and 5 and the other group consists of the nodes 3, 6 and 7. With respect to this assumption, the mobility based DGA algorithm must identify these two distinct groups of nodes, so that the hierarchy generated is robust. Furthermore, for this example it is assumed that the metric of interest is the relative velocity, since the grouping of nodes with similar mobility characteristics is suggested from the hierarchy generation objectives. In *Phase I* the nodes broadcast their node IDs, their direction and speed to their 1-hop neighbors. Also, at the same time they collect the corresponding information (node ID, direction and speed) from their 1-hop neighbors.

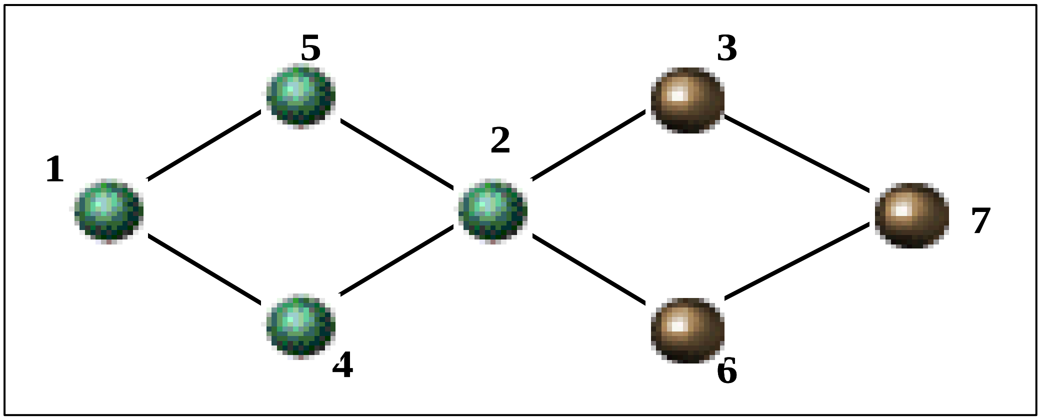


Figure 6.1. Domain generation example: Sample Network

After having collected the appropriate information, the participating nodes compute their relative velocity with each one of their 1-hop neighbors. Even though the relative velocity is computed locally to each node, the value computed from a pair of nodes (*i*,*j*) is the same, independently of whether is computed at node *i* or node *j*. Assume that for this example the relative velocities computed for each pair of 1-hop neighbors are provided from figure 6.2 below:

Relative Velocity

$U_{r_{ij}} = U_{r_{ji}}$

node ID	1	2	3	4	5	6
2	-					
3	-	2				
4	0.2	0.3	-			
5	0	0.1	-	-		
6	-	2.5	-	-	-	
7	-	-	0.1	-	-	0

Figure 6.2. Relative velocity values as computed pairwise from neighboring nodes

Based on the above values of relative velocity, each node selects the most dominant one hop neighbor (e.g. lowest relative velocity - the lower the relative velocity, the more similar the mobility) with whom he wants to group with for the formation of a domain Following the selection rule of this example in combination with the values provided in figure 6.2, the neighbor selections of the participating nodes by the end of *Phase I* are:

$$1 \rightarrow (5) \quad 2 \rightarrow (5) \quad 3 \rightarrow (7) \quad 4 \rightarrow (1)$$
$$5 \rightarrow (1) \quad 6 \rightarrow (7) \quad 7 \rightarrow (6)$$

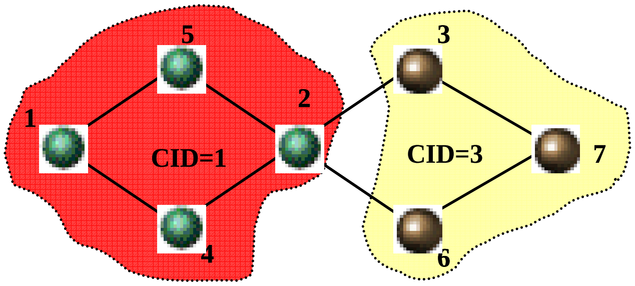
By entering *Phase II*, the nodes communicate their decisions from *Phase I* to the 1-hop neighbors they have selected. Each node collects and records the IDs of the nodes from which they have been selected into the *SelectedFromList*. The *SelectedFromList* for each one of the participating nodes is:

$$1 \leftarrow (4,5) \quad 2 \leftarrow (\;) \quad 3 \leftarrow (\;) \quad 4 \leftarrow (\;)$$
$$5 \leftarrow (1,2) \quad 6 \leftarrow (7) \quad 7 \leftarrow (3,6)$$

Each participating node, by combining the *SelectedFromList*, the selection made in *Phase I* and his own ID, forms the *InfoExchange* list. The *InfoExchange* lists of the nodes by the end of *Phase II* are:

$$\mathit{InfoExchange\ Lists}$$
$$1:(1,4,5) \quad 2:(2,5) \quad 3:(3,7) \quad 4:(1,4)$$
$$5:(1,2,5) \quad 6:(6,7) \quad 7:(3,6,7)$$

Phase III completes the generation of the hierarchical structure. Each node will utilize the *InfoExchange* list in order to communicate with its neighbor nodes and ultimately select the domain to join. The nodes are listening to the domain selections of the lower ID 1-hop neighbors that belong to their *InfoExchange* list until their turn comes to decide on the domain to join. After they decide, they wait for the rest of the 1-hop neighbors (e.g., nodes with higher IDs) in their *InfoExchange* list to decide. For the specific network of figure 6.1, the generated hierarchical structure after the completion of *Phase III* looks like:



$$CID = 1:\{1,2,4,5\}$$
$$CID = 2:\{3,6,7\}$$

Figure 6.3. The hierarchical structure established from the mobility based DGA algorithm

If we carefully observe, the values of the relative velocities of the nodes assigned from the distributed algorithm into the same domain, these values are much lower compared to the values of the relative velocities of the nodes assigned to different domains. The mobility based domain generation algorithm behaves in accordance to the suggested hierarchy generation objectives by grouping together the nodes with similar mobility characteristics. By doing so, the algorithm succeeds on generating robust to mobility domains, since it manage to identify the two groups of nodes with similar mobility. The nodes in the same domain are expected to remain connected for longer periods of time compared to the nodes that do not belong into the same group. As it is shown later in the performance evaluation section, the generated domains are more robust to topology changes compared to other distributed domain generation algorithms that do not take into consideration the dynamics of the network.

6.4 Performance Evaluation

This section elaborates on the ability of the proposed distributed domain generation algorithm to establish a hierarchical structure that is robust to mobility. The effectiveness of the approach is evaluated by comparing it with a well known distributed domain generation algorithm (lowest ID – LID) which forms domains without taking into consideration the dynamics of the network environment. The purpose of this comparison is to remark the importance of having the domain generation algorithm being aware of the network environment it operates into, since it can provide more stable hierarchical structure, which can be beneficial to the network instead of harmful.

6.4.1 Robustness of the mobility based DGA

The robustness of the mobility based DGA is measured from the stability of the domains' membership with respect to the mobility of the participating nodes. In order to highlight the effectiveness of the proposed algorithm we are comparing it with the lowest-ID (LID) algorithm [1], which utilizes metrics (the domain construction is based on the unique IDs of the nodes) unrelated to the network environment for the establishment of hierarchical structures.

LID selects cluster heads (CHs) among the participating nodes based on their IDs. For the domain formation, the remaining nodes (non-clusterhead nodes) are assigned to the CH node with the lowest ID among the CH nodes which are at most 1-hop away. The LID algorithm does not take into consideration the dynamics of the network for the domain formation since the selection metrics (proximity, nodes' ID) are independent from them.

We compared the membership stability of the domains obtained from the mobility based DGA with the corresponding stability of the domains obtained from the lowest-ID (LID) algorithm. To evaluate the robustness of the hierarchical structures generated by the mobility based DGA compared to LID we measured the average membership changes and the average number of generated domains. We applied both algorithms in network environments of various sizes and mobility levels. Namely, we generated networks of 100 to 1000 nodes and we applied the Random Waypoint Mobility (RWPM) model with pause time 0.

In the RWPM model each node selects a destination in the limits of the pre-specified area. This destination is approached with constant speed selected from the node at random. When the destination is reached the node selects new destination and new speed and the process is repeated. We investigated several scenarios corresponding to different maximum allowable speeds (between 1m/s and 10m/s) in order to evaluate the robustness of the generated hierarchies in various levels of mobility. We ran the algorithms (mobility based DGA, LID) for 1000s of network time. The statistics were sampled every 1s of network time.

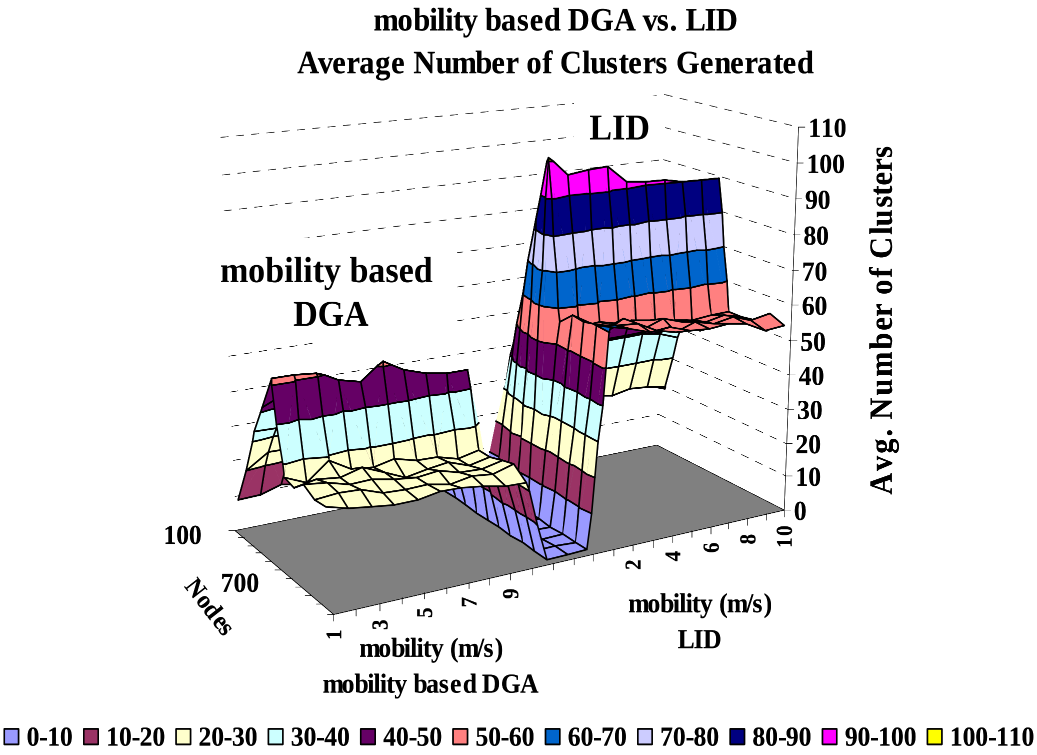


Figure 6.4. Average membership changes (LID vs. mobility based DGA) with respect to network size and mobility level.

Figure 6.4 represents the average membership changes for networks of size 100 to 1000 nodes and maximum allowable node speeds of 1m/s to 10m/s. The left part of the graph represents the average membership changes for the mobility based DGA algorithm and the right part represents the average membership changes for the LID algorithm, respectively. The higher the mobility and the larger the size of the network, the better the performance of mobility-based DGA algorithm compared to the performance of the LID algorithm. For example for 1000 nodes and 10m/s maximum speed, mobility-based DGA requires on average 32 membership changes per second and LID requires on average 44 membership changes per second. For 1000 seconds of network time, the mobility based DGA requires on average 12000 less membership changes than LID algorithm, which is an improvement of 27.2%.

Apart from the membership changes, a metric that indirectly characterizes the robustness of the proposed algorithm, is the average number of generated domains. The smaller the number of generated domains, the more tolerant is the hierarchical structure to the topological changes due to mobility (e.g. the larger the domain, the higher the probability of a node, whose connectivity changes, to remain connected to its original domain).

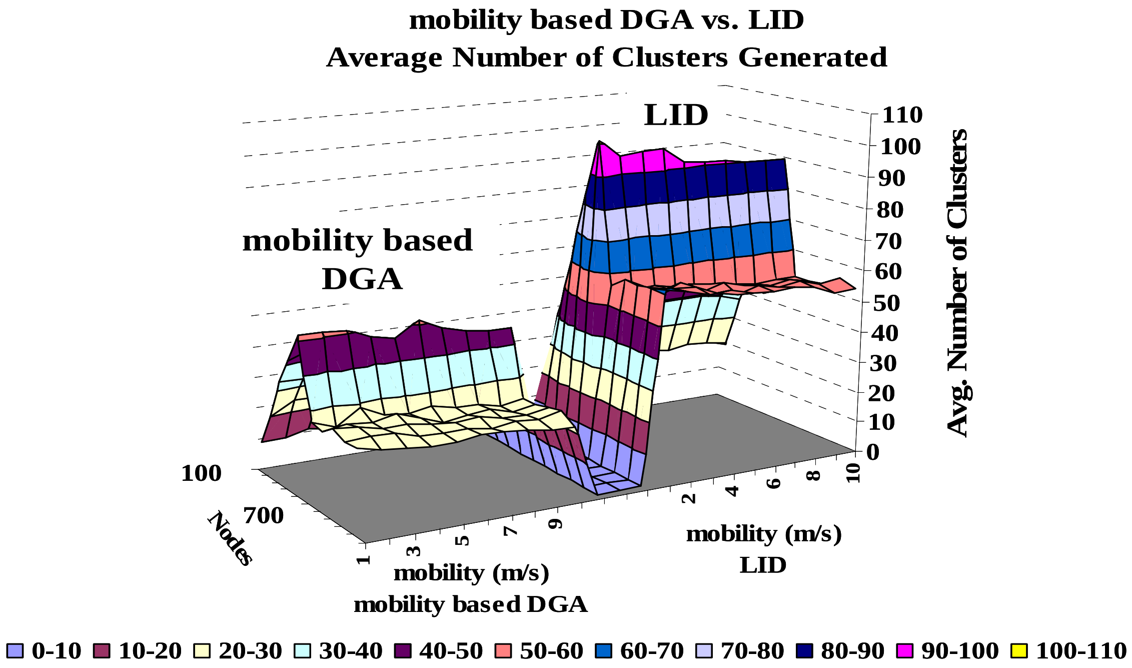


Figure 6.5. Average number of domains generated from LID and mobility based DGA algorithms for various network sizes and mobility levels

Figure 6.5 demonstrates the average number of clusters generated from each one of the algorithms. The left part of the graph represents the average number of domains generated from the mobility based DGA and the right part represents the average number of domains generated from the LID algorithm, respectively. The general observation is that the number of domains that LID generates is more than double the number of domains generated from the proposed algorithm. This observation suggests that the average domain size of the mobility based DGA is more than double the average domain size of LID. The latter can also be explained from the fact that LID does not generate domains with diameter larger than 2-hops as opposed to mobility DGA, which does not have such restrictions.

Because LID generates domains without taking into consideration the network environment, it is more possible to harm the performance of the network. The larger the number of membership changes, the larger the introduced overhead to the applied protocols. The stability of the hierarchy generated from the mobility based DGA aims on the minimization of this overhead, so that the overall performance of the network is improved.

Even though the superiority of the proposed algorithm has been presented with respect to the RWPM model, the stability of the hierarchical structures established from the mobility based DGA is expected to be even better in scenarios where group mobility is exploited. If we evaluate the proposed algorithm with respect to a group mobility model (Reference Point Group Mobility model), it is expected to establish more robust hierarchical structures compared to RWPM. The mobility based DGA was designed to identify and group together nodes with similar mobility characteristics, so in a network environment, where distinct mobility groups exist, the algorithm tends to eliminate the overhead due to the membership changes by accurately grouping together the nodes that present similar mobility characteristics.

Chapter 7: Domain Maintenance Approaches

7.1 Introduction

Domain autoconfiguration techniques allow the quick formation of highly optimized hierarchies that greatly enhance network scalability and overall performance. For example, instead of producing a simple two level hierarchy based only on topology, the optimization can produce multi-level hierarchies that take into account factors such as mission goals and predicted node/link heterogeneity. However, in dynamic networks, these highly optimized solutions degrade very quickly. Indeed, the application of standard local maintenance algorithms that do not align well with the optimization goals, may result in very fast degradation of the hierarchical structure's quality (cost) with respect to the hierarchy generation objectives. In this chapter a new taxonomy of local maintenance algorithms into four basic classes is being presented. Furthermore, the performance benefits for using representative approaches that act in accordance with the optimization goals are being quantified. The classification of the local maintenance algorithms and the quantification of their effect on the maintained hierarchical structure can be proved beneficial on their design. The main trade off between the various categories of local maintenance mechanisms is the amount and quality of information available and the ability to preserve the quality of hierarchical structure with respect to the hierarchy generation objectives.

The mobile ad hoc networks (MANETs) require no fixed infrastructure, making them ideal for many commercial, emergency and military scenarios. An open question, however, is the ability of MANET networks to scale. Even for protocols (routing, security, QoS) designed specifically for these dynamic environments, when the size of the network becomes too large then these protocols either fail to capture the network dynamics or swamp the network in signaling traffic [17][21][22][38][40].

Although some protocols can scale to hundreds or even thousands of nodes in certain conditions, in general network scalability has always relied on the generation of hierarchy. For example, the wireline world divides networks into subnets and Autonomous Systems. The affect of hierarchy can be dramatic. For example, in theory, clustering can reduce the overall routing protocol overhead with n nodes from $O(n^2)$ to $O(n\log n)$. Network hierarchy allows the applied protocols to operate on smaller subgroups of the network and not on the entire network. Their hierarchy allows protocols to deal with the dynamics of smaller groups of nodes. Hierarchy also allows protocols to be tuned to more homogenous conditions. The benefits of a good hierarchy have been shown to outweigh the complexity [39].

In order to cope with the rapid deployment and rapid reconfiguration needed for future versatile networks, the generation of hierarchical structure must be done automatically. Moreover, in mobile ad hoc networks (MANETs), such as the FCS (in a Unit of Action) or WIN-T (in a Unit of Employment), there is a need for mechanisms that not only automatically create such hierarchies but also maintain them as the network changes.

In the next section the various local maintenance approaches are categorized and the characteristics of each of the defined classes are being presented. In section 3 we evaluate the effect of the various local maintenance approaches on the preservation of the quality (cost) of the optimized generated hierarchy with respect to the set of hierarchy generation objectives. Finally, section 4 highlights the most important conclusions obtained from the taxonomy and study of the various classes of local maintenance mechanisms. The conclusions of this study can be utilized as a blueprint for the design of appropriate local maintenance algorithms to support the application of hierarchy generation mechanisms in dynamic environments.

7.2 Hierarchy Maintenance Schemes

Even though, the introduced domain generation framework is capable of generating optimized hierarchical structures, due to the dynamic nature of the underlying network environment, these structures will soon become sub-optimal and might not comply with the topological constraints (e.g. infeasible hierarchical structures). It would be inefficient and expensive to apply the hierarchy generation mechanism for every topological change happening into the network. Firstly, the hierarchy generation mechanism involves all the nodes, so extra network resources will have to be utilized frequently causing large scale reactions from the nodes with possible negative impact on the network performance. Secondly if the SA-based generation algorithm is to be used, global knowledge is required, which may impossible to be obtained and processed on time between each topological change, especially in highly dynamic networks. Thus, the maintenance of the hierarchical structure in case of topological changes must involve localized information for the reconstruction of the domains in a distributed fashion. The objectives for designing and applying a distributed hierarchy maintenance approach, instead of utilizing the introduced hierarchy generation mechanisms are the:

- Reduced overhead,
- Faster hierarchy reconstruction

Even though the distributed maintenance approach will be beneficial with respect to network resources and reaction time to the dynamic network changes, considering the spirit of the hierarchy generation framework proposed, the domains have to satisfy a set of objectives, so that the hierarchy can be beneficial to the network performance. The quality of the generated hierarchy is expressed through its cost computed with respect to the cost function representing the generation objectives. These objectives have to be preserved after the application of the maintenance algorithm so that the quality of the hierarchical structure is maintained throughout the lifetime of the network. Such a task is very challenging due to the limited availability of information (localized information).

In this chapter the various local maintenance techniques have been categorized with respect to the amount of information available and the relevance of this information to the hierarchy generation objectives. Furthermore the impact of the defined local hierarchy maintenance classes on the preservation of the hierarchical structures “quality” (cost) has been studied and some very interesting observations are being presented.

7.3 Taxonomy of Local Maintenance Schemes

The main trade off in distributed maintenance is the tradeoff between overhead and hierarchy quality. Four classes of local maintenance approaches have been identified:

- **A0**: Zero Overhead Local Maintenance
- **A1**: Objectives Independent Local Maintenance
- **A2**: Node Dependent Local Maintenance
- **A3**: Domain Dependent Local Maintenance

Figure 7.1 describes the classification of the various approaches introduced in this study with respect to the amount and quality (e.g. relevance with the generation objectives) of information involved in their decision mechanisms.

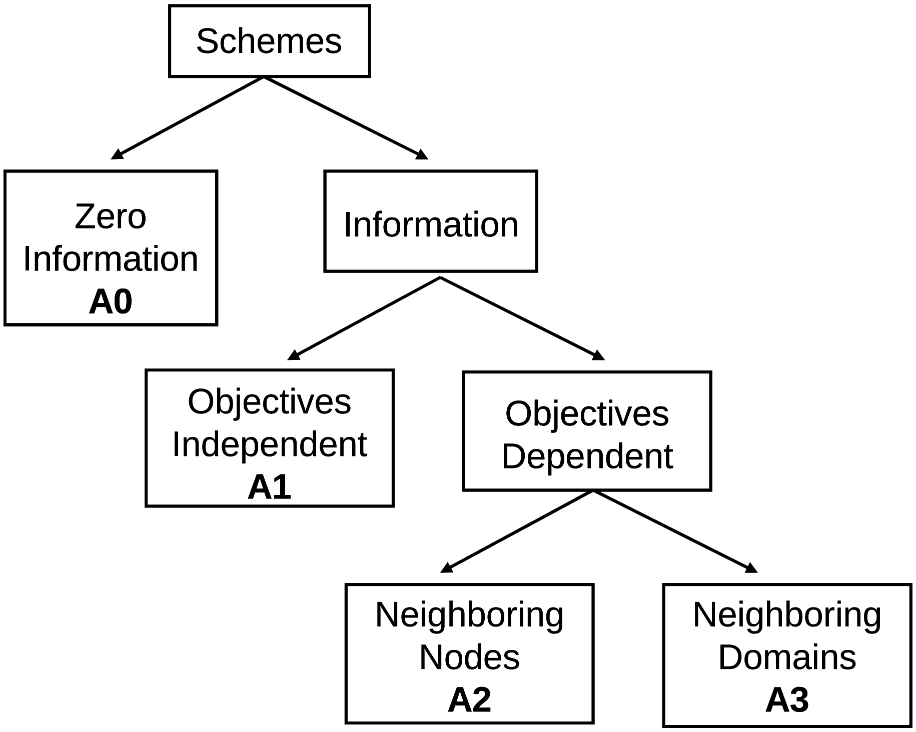


Figure 7.1. Taxonomy of local maintenance approaches

In general, it is expected that the more relevant to the generation objectives (higher quality) the information available during the maintenance phase, the better is preserved the "quality" of the hierarchical structure. The maintenance decisions will be based on metrics related to the original hierarchy generation objectives. For example if the hierarchy generation objective is the construction of robust to mobility domains, then the local maintenance is better to utilize metrics related to the speed, direction and position of the participating nodes for the reconstruction of the hierarchy.

In general the maintenance method is triggered locally by the nodes that become infeasible (e.g. the nodes lose connectivity to their original clusters) due to the topological changes. An overview of the amount and quality of information required from each class of algorithms is provided below:

- **A0. Zero Information Local Maintenance** This approach does not require any information to be collected from the network for the reconstruction of the hierarchical structure. The approach in terms of overhead is optimal, since it does not utilize any bandwidth resources but the lack of information is expected to result in poor preservation of the hierarchical structure's cost.

- **A1. Objectives Independent Local Maintenance** The schemes of this approach collect and utilize local information for the reconstruction of the hierarchical structure. The information, however, is unrelated to the metrics that have been utilized from the hierarchy generation mechanism for the construction of the optimized hierarchical structure. For example when the generation objectives enforce the formation of robust to mobility domains, the speed and direction of nodes is required so that are grouped based on their mobility similarities. To the contrary, during the maintenance the nodes may have access only to information independent of the mobility characteristics of the neighboring nodes (i.e. IDs of the neighboring nodes).

- **A2. Node Dependent Local Maintenance.** The approach (A2), as opposed to the previous two, is aware of the hierarchy generation objectives and the corresponding schemes attempt to maintain the “quality” of the generated hierarchy by utilizing metrics related to these objectives. However, the maintenance decisions are based on information gathered only from the immediate neighbors (e.g. one hop neighbors).
- **A3. Domain Dependent Local Maintenance.** Like scheme (A2), A3 utilizes relevant, to the hierarchy generation objectives, information for maintaining the “quality” of the hierarchical structure; but unlike A2, A3 bases its restructuring decisions on information collected from the entire neighboring domains. Clearly, this approach requires the most overhead but it is expected to have the more beneficial impact on the maintenance of the hierarchical structure’s quality.

7.4 Local Maintenance Representative Schemes

This section provides representative schemes from each of the four hierarchy maintenance classes. These schemes will be able to provide a better insight on the characteristics of the local maintenance algorithms of each of the hierarchy classes that constitute the taxonomy defined above.

- **A0. Random**. As its name reveals, the random maintenance mechanism is probabilistic. Specifically, the nodes seeking to join a domain, randomly select one of available neighboring domains with respect to the uniform distribution. If V^k is the set of neighboring domains C_i of node k defined as:

$$V^k = \left\{ C_i : \exists j \in C_i \text{ s.t. } j \xleftrightarrow{\text{1 hop}} k \right\}$$

then node k selects a domain C_i with probability $p_k\left(C_i\right)$, where

$$p_k\left(C_i\right) = p_k = \frac{1}{\left|V^k\right|} \tag{7.1}$$

Given the selection probabilities (7.1), node k generates a random number $r_k \sim U\left[0,1\right)$, which defines the neighboring domain C_z to join as follows:

$$C_z = \left\{ z \in \left[\mathbb{Z}^+ \cap \left[1, \left|V^k\right|\right]\right] : \left(z-1\right)\cdot p_k \le r_k < z \cdot p_k \right\} \tag{7.2}$$

This scheme belongs in class A0 since it obtains the maintenance decisions probabilistically, so the knowledge of any metric related to the characteristics of neighboring nodes is not required.

- **A1. Lowest ID (LID)**. The lowest ID (LID) scheme requires that each node owns a unique ID. The lowest ID node among the nodes of each domain C_i determines the ID of this domain. The latter domain naming approach results into conflict free naming of the generated domains due to the uniqueness assumption of the node IDs. When a node k seeks to join a new domain, it selects the lowest ID domain C_z among the set V^k of its neighboring domains.

$$C_z \triangleq \underset{C_z \in V^k}{\arg\min}\left(ID\left(C_z\right)\right) \tag{7.3}$$

where,

$ID\left(C_i\right)$: is the ID of the i^{th} cluster C_i

This scheme belongs to class A1 of local maintenance schemes due to the utilization of information (unique node ID) that is not related to any of the hierarchy generation objectives for the network performance improvement.

- **A2. Node Dependent Cost Function**. This scheme relies on metrics relevant to the objectives of the hierarchy generation phase. Even though the metrics are relevant, their availability is limited. Each node is aware of the value of these metrics from its immediate (one hop) neighboring nodes only. For example if the hierarchy generation objective is to construct robust to mobility domains by grouping together nodes with similar mobility characteristics (speed, direction); a node, which relies on a maintenance scheme of A2 class, seeking to join a new domain; will join the same domain as its neighboring node with the more similar mobility characteristics (speed, direction). In cases where multiple choices exist (multiple neighbors present the same degree of metrics similarity), a tiebreaker rule is applied (i.e., the node joins the domain of the similar neighbor with the lower ID).

- **A3. Domain Dependent Cost Function**. This scheme, similarly to the previous scheme, uses relevant metrics to the objectives enforced during the hierarchy generation phase. Whereas, the availability of these metrics is less limited compared to the schemes of class A2, since the metrics are being collected from the entire neighboring domains and not just from the immediate neighboring nodes. The selection of the domain to join among the candidate domains happens by computing locally the same cost function utilized in the hierarchy generation phase, for each one of the candidate selections. The selection which results into the local domain configuration with the best cost becomes the dominant selection of the node. For example, assume that the hierarchy generation objective is the construction of domains robust to mobility and is represented from the following cost function:

$$J\left(C\right)=\min_{C}\left(\sum_{z=1}^{K}\left[\sum_{i,j=1}^{\left|C_z\right|}U_{r_{i,j}^{z}}^{2}\right]^{2}\right) \qquad (7.4)$$

where,

Parameter	Definition
K	Number of generated clusters
C_i	Cluster *i*
$\left\|C_i\right\|$	Size of cluster *i*
$U_{r_{i,j}}$	Relative Velocity of nodes *i*, *j*

A node i, which utilizes a local maintenance scheme of class A3 and seeks to join a neighboring domain, will collect the appropriate metrics from all the nodes of the neighboring domains. Then for each possible local domain configuration it will compute cost function (7.4) by utilizing the metrics collected and its own metrics. The selection that results to the lowest cost domain configuration will become the dominant configuration.

The following section provides an indicative example on the application of the representative local maintenance schemes described in this section and their effect on the maintenance of the hierarchical structure quality, with respect to the applied cost function.

7.5 Sample Application and Indicative Performance of the Representative Local Maintenance Schemes

For providing both a basic understanding of the local maintenance schemes' functionality and a representative view of their performance subject to the preservation of the hierarchical structure's quality, this section utilizes one of the introduced cost functions and the corresponding metrics for the generation of hierarchy. Then, each of the local maintenance schemes is applied appropriately and the cost of the maintained structure is computed and compared to the optimal.

7.5.1 Representative Hierarchy Generation Objective

Consider as hierarchy generation objective the construction of robust to mobility domains by grouping together nodes of similar mobility characteristics. In the hierarchy generation phase the domains are formed by optimizing the cost function (7.2) using the SA algorithm.

$$J\left(C\right) = \min_{C}\left(\sum_{z=1}^{K} \left[\sum_{i,j=1}^{|C_z|} U^{2}_{r^{z}_{i,j}} \right]^{2} \right) \tag{7.5}$$

where,

C_i :	Cluster *i*
$\left|C_i\right|$:	Size of cluster *i*
$U_{r_{i,j}}$:	Relative Velocity of nodes *i*,*j*

The relative velocity $U_{r_{i,j}}$ of two nodes i, j is defined from (7.6), (7.7) and (7.8).

$$U_{r_{i,j}} = \sqrt{U_{X_{i,j}}^2 + U_{Y_{i,j}}^2} \tag{7.6}$$

$$U_{X_{i,j}} = S_i \cos\theta_i - S_j \cos\theta_j \tag{7.7}$$

$$U_{Y_{i,j}} = S_i \sin\theta_i - S_j \sin\theta_j \tag{7.8}$$

where,

S_i: Speed of node *i*

θ_i: Direction of node *i*

Assume the network and optimized hierarchy of Figure 7.. Due to mobility, node 11 changes the topological structure of the network and seeks to join a neighboring domain. Such an event triggers the maintenance phase. The representative schemes introduced above are being applied so that their impact on the “quality” (cost) of the maintained hierarchical structure can be evaluated.

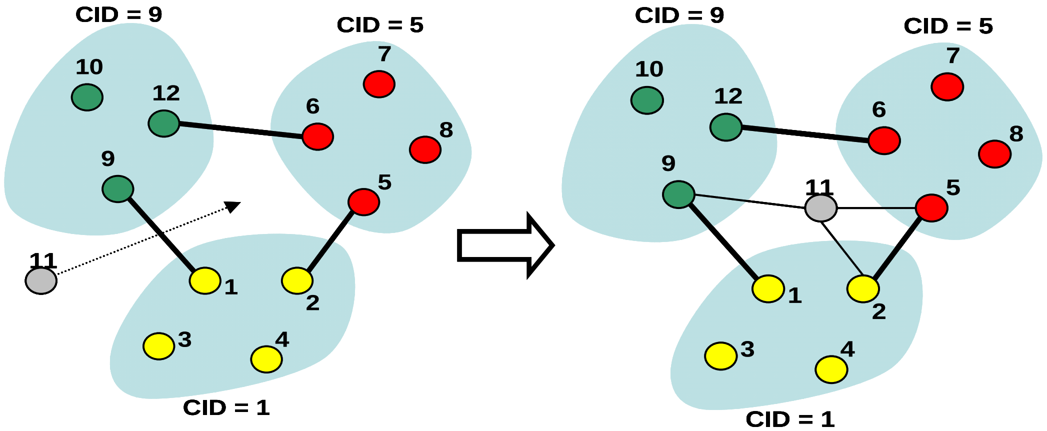


Figure 7.2. Topological change triggering the application of local maintenance

Assume also that the mobility metrics - speed (Sp) and direction (Dr) - of the nodes are a given in Table 7.1.

ID	Sp	Dr	ID	Sp	Dr	ID	Sp	Dr
1	0	0	5	4	45	9	3	45
2	0	0	6	5	60	10	3	30
3	0	0	7	5	45	11	4	45
4	0	0	8	6	60	12	2	30

Table 7.1. Mobility characteristics of the nodes

7.5.2 Application of the Local Maintenance Schemes

Figure 7.3 presents the domain selections made by Node 11 (from Figure 7.) by applying each one of the representative schemes:

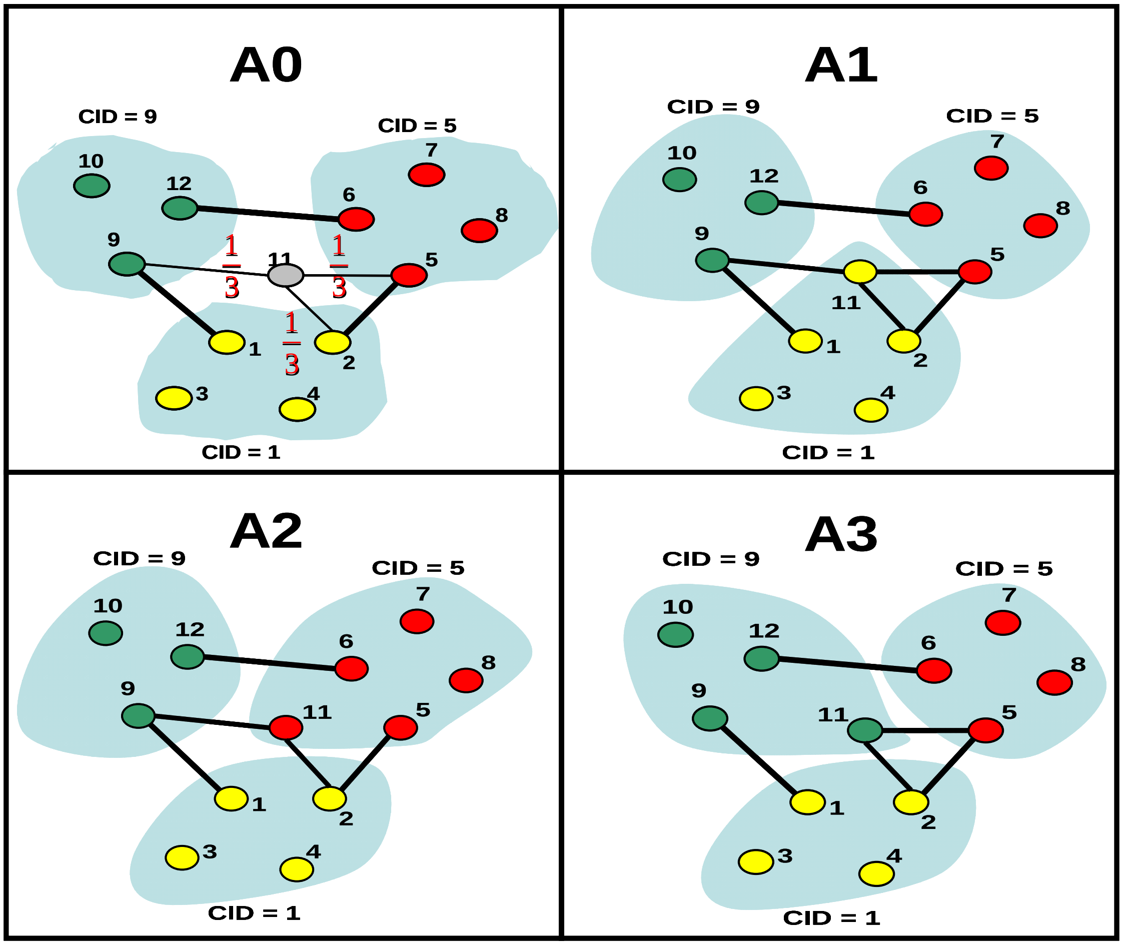


Figure 7.3. Hierarchy resulted by the application of the representative schemes

The selection process followed by Node 11 by applying each one of the local maintenance schemes is described below:

- **A0 (Random)**: Node 11 detects three neighboring domains, and will decide to join randomly one of these with respect to uniform distribution. The probabilities of Node 11 (p_{11}) for joining any of the neighboring domains C_i are:

$$p_{11}(C_1) = p_{11}(C_2) = p_{11}(C_3) = \frac{1}{3}$$

So, Node 11 can equiprobably select each one of the neighboring domains to join, resulting into different hierarchy configuration costs, as it is shown later

- **A1 (Lowest ID)**: Node 11 will decide to join domain C_1 because it has the lowest ID among its neighboring domains:

$$C_1 = 1, C_2 = 5, C_3 = 9$$

- **A2 (Node Dependent Cost Function)**: With respect to speed and direction values given in Table 1, Node 11 has speed 4m/s and direction of 45 degrees. The neighboring nodes of Node 11 and their corresponding domains are represented from the following (node ID, domain ID) pairs:

$$(2, C_1 = 1), (5, C_2 = 5), (9, C_3 = 9)$$

Node 5 has the closest match in mobility with Node 11. Thus Node 11 will select to join the same domain as of Node 5$(C_2 = 5)$.

- **A3 (Domain Dependent Cost Function)**: Node 11 collects the appropriate metrics (e.g. speed and direction) from each one of the nodes lying in its neighboring domains. Using function (7.5), Node 11 evaluates the cost of the resulted maintained structure in the case of joining each of its neighboring domains. The resulted costs are provided from the following (domain ID, cost) pairs:

$$\left(C_1 = 1,\ 81.77\right), \left(C_2 = 5,\ 26.77\right), \left(C_3 = 9,\ 25.13\right)$$

Thus Node 11 will pick to join domain $\left(C_3 = 9\right)$, which results in the hierarchical structure with the lowest cost among the various choices.

7.5.3 Cost Performance Comparison of the 4 Local Maintenance Approaches

For this example, each scheme results in a different hierarchical structure with different cost ("quality"). Table 7.2 below reveals the cost of the maintained hierarchy for each scheme applied in this example.

Approach	**Cost**
A1. Objectives Independent (LID)	C_{A1}= 81.7689
A2. Node Dependent (A2)	C_{A2}= 26.7673
A3. Domain Dependent (A3)	C_{A3}= 25.1318
A0. Zero Information (Random)	$C_{A1} \vee C_{A2} \vee C_{A3}$

Table 7.2. Cost of the hierarchy after the application of the various Local Maintenance schemes

A couple of indicative and very important observations are:

- The lowest cost (best hierarchy) is provided from approach A3 scheme. The application of approach A3 schemes is expected to perform best, because it takes into consideration metrics from the entire neighboring domains, which are also relevant to the generation objectives. The drawback of the maintenance schemes of this approach is the large overhead introduced for the collection of the appropriate information. Whereas, the "quality" of the maintained hierarchy compensates for this drawback.

- Even though, the Random scheme of approach (A0) uses no metrics for the selection process (zero overhead), it is statistically expected to perform better than the schemes of approach A1, such as the LID, with respect to the "quality" of the maintained hierarchy.

7.6 Impact of Maintenance Schemes on Domain Quality

This section shows that the impact of the schemes on the cost of the hierarchy for the specific example given in the previous section is representative of the generally expected performance of the corresponding approaches. From the cost functions introduced in chapter 4, two of them have been used to construct optimized hierarchies. Then, for a pre-specified amount of time, the various local maintenance schemes have been applied. The cost of the maintained hierarchy was being evaluated in pre-specified intervals of time, so that samples of the maintained hierarchy's cost can be obtained throughout the application of each of the schemes. These samples are sufficient to characterize the impact of the each of the schemes and their representation classes on the maintained hierarchy quality.

7.6.1 Impact of Schemes on "Balanced Size" Domains

On a network of 100 nodes 10 domains were defined utilizing the hierarchy generation mechanism based on the SA algorithm. The hierarchy was generated with respect to the "balanced size domains" objective. By optimizing (minimizing) the following cost function (see chapter 4):

$$J(C) = \min_{C}\left(Var\left(\left|C_1\right|^2,....,\left|C_K\right|^2\right)\right) \tag{7.9}$$

10 domains of 10 nodes each were obtained. Then, for 500 seconds of network time, the representative maintenance schemes of approaches (A0), (A1) and (A3) were applied on the optimized hierarchy.

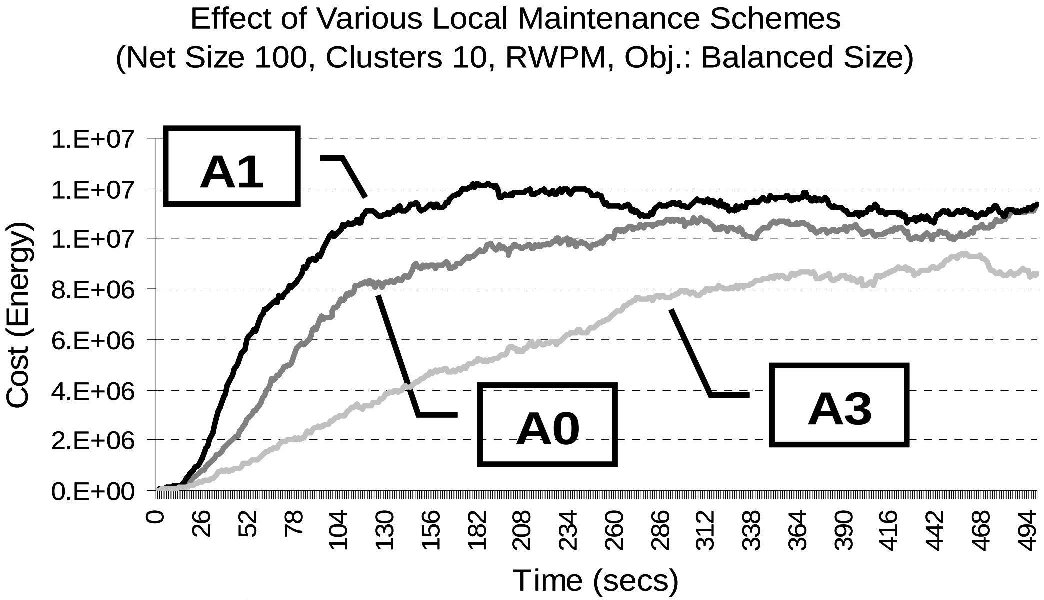


Figure 7.4. Impact of three maintenance approaches on the "balanced size" domains

Figure 7.3 shows the average cost per second (out of 100 applications) of the maintained hierarchy. The topology was changing every second with respect to Random Waypoint Mobility (**RWPM**) model, with maximum speed 10m/s and no pause time. For every network second, the cost of the maintained hierarchy was evaluated using cost function (equation 7.9).

As expected scheme (A3) performs the best (but also causes more overhead). Interestingly, the Random scheme (A0) maintains better the "quality" of the hierarchy than the LID scheme (A1), even though does not collect and utilize any information from the neighboring nodes or domains (e.g. does not require any overhead).

7.6.2 Impact of Schemes to "Robust to Mobility" Domains

A second experiment generated 6 domains in a network of 100 nodes. This time the cost function (equation 7.5) was applied, resulting into the grouping of nodes with similar mobility characteristics. After obtaining the optimized hierarchical structure, each of the introduced local maintenance schemes, representing the four maintenance classes, were applied for 250 seconds. The topology of the network was changing every second with respect to Reference Point Group Mobility (RPGM) model [66] (6 groups of nodes with distinctive mobility characteristics were predefined, so the cost function applied had to locate these 6 mobility groups – optimal grouping). Figure 7.5 presents the average cost of the maintained hierarchy per second (out of 100 applications) for the various maintenance schemes.

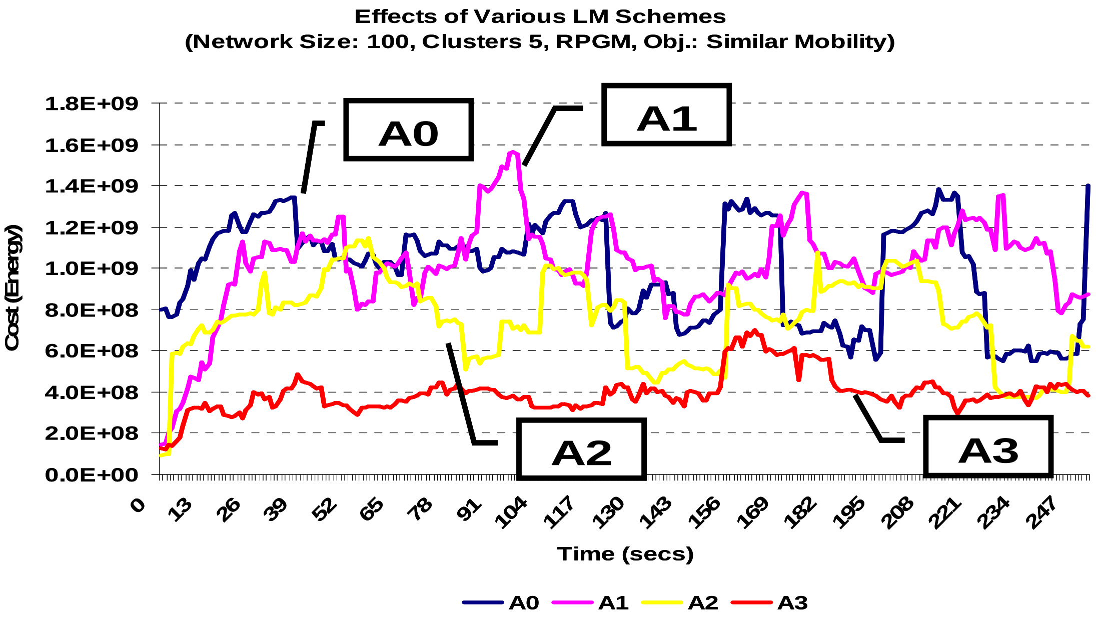


Figure 7.5. Impact of maintenance approaches on the "robust to mobility" domains

Similarly to previous scenario (e.g. “balanced size” domains), approach (A0) performs better than (A1), but both of them perform worse compared to (A3). Also, by comparing (A2) with (A3), (A3) performs better (as expected due to the larger amount of information it utilizes for the maintenance decisions).

7.7 Conclusions

This chapter categorizes various local maintenance approaches, with respect to: a) whether they are aware of the hierarchy generation objectives utilized during the generation phase, and b) the amount of information available to them. We show that by ignoring the local maintenance algorithm, hierarchy generation may end up harming the performance of the network instead of improving it. The maintenance algorithm has to be designed in accordance to the performance objectives required. The most commonly used approach applied today, the Lowest ID approach (A1), consistently performs the worst. Better for both quality and overhead is a Random Approach (A0). However, tailoring the maintenance to the hierarchy generation objectives consistently maintains the best quality hierarchy, even though extra overhead is required for the collection of the appropriate information. It is shown, for example, how simple a balanced size hierarchy can be maintained over 5x longer, while more complex hierarchies, that take into account factors such as mobility, can be maintained over 100x longer. This longevity is critical to maintaining the sort of effective and powerful network needed relying only on local information and decisions. In this fashion the reapplication interval of the hierarchy generation mechanism is prolonged, resulting in less overhead and faster reaction to the network dynamics.

Chapter 8: Network Applications of Hierarchy Generation Mechanisms

8.1 Introduction

The hierarchy generation and maintenance framework being developed has been tuned and evaluated, using a set of indicative cost functions. These cost functions represent hierarchy generation objectives, which are not directly related to any of the networking functions (e.g. routing, security, QoS). The hierarchy generation and maintenance framework is not customized to specific objectives and aims on the improvement of the network performance. To achieve the latter, the appropriate cost functions have to be introduced and optimized. These cost functions will represent the performance aspects of the network required to be improved.

In order to apply and evaluate the ability of the introduced mechanisms to construct hierarchical structures that take into consideration and improve realistic performance aspects of the networking functions, a new set of cost functions has been introduced. These cost functions are directly related to the networking functions and performance metrics, rather than focusing on the physical characteristics of the participating nodes and generated domains. Specifically in this dissertation, the SA-based framework has been applied for the generation of hierarchical structures aiming on the improvement of routing and topology control.

A direct application of the developed hierarchy generation framework is the reduction of power consumption due to transmissions. The transmission power is one of the dominant elements of the node's power consumption characteristics.

Specifically, the approach utilizes the SA-based mechanism to determine the optimal assignment of transmission power (transmission range) to each of the participating nodes such that the network is 2-d connected but also the average power utilized is as low as possible. The conservation of power in wireless environments that consist of finite power portable devices is crucial for the survivability and lifetime of the network. If this is not taken into account, nodes soon will start failing resulting in partitioned networks unable to provide services to the participating users. Moreover, by controlling the transmission power, the performance of MAC and subsequently of the network improves due to the lower number of collisions and retransmissions. The generation of power consumption aware hierarchical structures can be beneficial for the connectivity, survivability and lifetime of the network. Section 2 presents the details of the approach along with the metrics and cost functions introduced for the generation of the corresponding hierarchical structures.

Furthermore, another direct application of the hierarchy generation framework is on the improvement of hierarchical routing. The application of routing in a hierarchically organized network results into paths that are longer with respect to hop count than the paths that could be established in a flat network. This is the negative effect of abstraction resulting from the aggregation of information happening for each group of nodes. Due to the aggregation, the routing has an abstracted view of the network topology, which results in suboptimal path lengths. The latter suboptimality could potentially have serious consequences on the performance of the network. For example, the longer path lengths may harm the end-to-end delay performance and the throughput of the network (i.e. the transmitted packets will live longer into the network causing heavier load and more collisions in MAC, resulting in larger number of dropped packets). The cause of the suboptimality in most of the cases is that the hierarchy generation mechanism does not take into account this side-effect. Since the SA-based mechanism is powerful and flexible enough to combine and satisfy multiple objectives, the improvement of hierarchical routing path length suboptimality can be performed in conjunction with other objectives by optimizing the appropriate multi-objective cost functions. The details of the approach for the reduction of hierarchical routing path length suboptimality are provided in section 3.

8.2 Hierarchy Generation for Power Control and Connectivity Assurance

Ad hoc networks are the new networking technology trend due to their promising characteristics. These characteristics accommodate better the requirements imposed by the commercial and military world. Most of the reference applications for these networks assume devices that are of finite power. The validity of the latter assumption is justified from the characteristics of the existing technology (web enabled cell phones, PDAs, laptops, PPCs). The objective of the ad hoc networks existence is to accommodate light weight, battery powered portable devices. Because of the finite power limitation, the appropriate mechanisms have to be introduced for the efficient utilization of the available power. A first step towards this direction is the minimization of the unnecessary power utilization, wherever possible.

An active networking device consumes significant portion of its power for communications and more specifically for transmissions. Therefore, the proposed approach focuses on minimizing the average transmission power by adjusting appropriately the transmission powers of the participating entities subject to the preservation of network connectivity. The network connectivity requirement is essential since the utilization of lower transmission power from the participating entities could result in partitioned networks. The enforcement of the latter requirement (constraint) in the SA-based optimization process will control the achievable minimum average power assignments that result in connected networks. The fundamental principal of the mechanism introduced for optimal transmission power assignments, is the grouping of the participating entities with respect to their topological proximity. After the grouping has occurred, the intra-domain and inter-domain connectivity of the network is ensured from the application of the appropriate heuristic mechanisms. Thus, both network connectivity and optimization of the average node transmission power assignments are achieved.

The development of the mechanisms is based on the assumption that the transmitted power utilized from the nodes is proportional to the transmission range. The nodes are grouped with respect to their relative proximity characteristics since it is expected that the members of the same domain communicate with each other on a regular basis (e.g. intra-cluster communication for control signaling). Hence, it is preferable these members to be topologically close. In the case of the existence of information that requires inter-cluster communication, dedicated nodes (Border Routers - BRs) are responsible for the forwarding of this information through the hierarchy until it reaches the destination domain. For ensuring the inter-cluster connectivity the assignment of the appropriate nodes as BRs is required. The BRs selection mechanism has also to be aware of the transmission power optimization (minimization). The connectivity of BRs in this architectural framework is sufficient for ensuring network connectivity, since the intra-cluster connectivity will have been established in an earlier stage of the mechanism.

For the grouping of nodes the introduced SA-based algorithm has been applied on a representative cost function, which is related to the topological proximity of the nodes. The metrics required for the evaluation and optimization of the corresponding cost function, are related to the topological positions of the nodes. This information can be collected from GPS devices, attached to the nodes and can be stored in an accessible centralized or distributed database. Furthermore, the same information is required from the heuristic algorithm responsible for the assignment of transmission range (transmission power) to the participating network entities. The heuristic algorithm is initially applied locally to each one of the generated domains, so that the average node transmission power is minimized subject to intra-domain connectivity. Moreover, for ensuring network connectivity, it is necessary that the generated domains are connected. Towards achieving inter-domain connectivity the set of candidate BRs has to be determined. Upon determining this set, a heuristic algorithm is applied for the selection of appropriate transmission ranges to the BRs, so that both inter-domain connectivity can be established and minimization of the average node transmission power can be accomplished. The description and characteristics of the heuristic algorithm are provided in a subsequent section. The following section surveys some of the most cited work done in the area of transmission range control.

8.2.1 Related Work on Transmission Range Control

In this section we refer to some representative work on the power control problem from the perspective of network clustering. The existing work problem comprises of techniques that try to identify the optimal transmit power to control the connectivity of the network. In [42], the power control problem is viewed as a network layer problem, and the COMPOW protocol is proposed. The work in [43] proposes that each node has to adjust its transmission power such that its connectivity degree (number of one-hop neighbors) is bounded. ElBatt et. al. in [41], through the transmit power control, make an attempt to optimize the average end-to-end throughput by controlling the degree of the nodes. In [44] a distributed topology control algorithm is proposed. The latter technique is based on the utilization of direction information. Kawadia and Kumar in [45] propose the CLUSTERPOW algorithm, which aims on the increase of network's capacity by increasing spatial reuse. The algorithm consists of simply using the lowest transmit power level p, such that the destination is reachable (in multiple hops) by using power levels no larger than p. The transmission range control mechanism proposed in this dissertation differs from the existing ones, since it is part of a more general hierarchy configuration framework. The same mechanism (SA) can assign transmission ranges to the participating entities and configure the network hierarchy at the same time. Such an approach has the advantages of simplicity, efficiency and robustness, since the need for interfacing among different mechanisms has been eliminated. The details of the transmission range control mechanism are provided in the following section.

8.2.2 Clustering and Transmission Range Control Algorithms

This section presents the algorithms and techniques that constitute the transmission range control mechanism for optimizing (minimizing) the average transmission power. The mechanism consists of four stages:

- Grouping of nodes
- Intra-cluster transmission range assignment
- Selection of BRs
- Inter-cluster transmission range assignment

The details for each of the four stages follow.

Stage 1: Grouping of Nodes

For the grouping of nodes with respect to their topological proximity, the SA-based framework has been applied for the optimization of the corresponding cost function. The metrics that constitute the cost function involve the topological coordinates of the nodes.. The objective is to group together nodes that are topologically near, since it is expected that this will reduce the average node transmission power (e.g., transmission range proportional to transmission power) for achieving domain and network connectivity. The corresponding cost function, which upon its optimization, will result in the grouping of the nodes with respect to their relative proximity is:

$$J\left(C\right)=\min_{C}\sum_{i=1}^{K}\sum_{j=1}^{\left|C_i\right|}\left\|x_{ij}-z_i\right\| \tag{8.1}$$

where,

C : Domains map

K : Number of generated clusters

$\left|C_i\right|$: The cardinality of *i*th cluster

x_{ij} : The coordinates of the *j*th node on the *i*th cluster

z_i : The center of mass coordinates of the *i*th cluster

Upon the optimization of the above cost function from the SA-based mechanism, a corresponding hierarchical structure will be generated. This stage is followed by the selection of the optimal transmission ranges to be assigned on each one of the nodes in the generated domains for establishing intra-domain connectivity.

Intra-Domain Transmission Range Assignment

After optimizing the grouping of nodes with respect to their topological proximity, the heuristic mechanism for assigning the optimal (minimum) transmission ranges to the nodes for the establishment of intra-domain connectivity takes over. The latter mechanism operates in each generated domain C_i separately and its objective is formulated from the cost function (8.2) below:

$$J\left(C_i\right) = \min \sum_{j=1}^{\left|C_i\right|} TxRange\left(node_{ij}\right) \tag{8.2}$$

subject to intra-domain connectivity

Since the above problem is *NP*-complete, for the optimization of (8.2), a faster suboptimal heuristic algorithm has been proposed instead. The objective of the algorithm is to assign to every node of a specific domain the lowest possible transmission range so that intra-domain connectivity is ensured. The details of the heuristic mechanism, applied to each domain C_k separately, are provided from the following pseudo code:

```
Step   I: Order the distances of each pair of nodes in the domain in ascending order. Store it in the
          vector SortedPairDistances
Step II:  Pick the first entry (lowest distance) from the SortedPairDistances and then store the
          corresponding pair of nodes in the ConnectedList list and the link in the LinksList list.
          Remove this entry from the  SortedPairDistances.
Step III: Until the $|ConnectedList| == |C_k|$
          Step IV: Pick the first entry – link $(\text{node}_i, \text{node}_j)$ from the SortedPairDistances such
          that one of the following is satisfied:
            1. $\text{node}_i \in ConnectedList$ & $\text{node}_j \notin ConnectedList$
            2. $\text{node}_i \notin ConnectedList$ & $\text{node}_j \in ConnectedList$
          Remove this entry from the SortedPairDistances
Step V:   Add the nodes $(\text{node}_i, \text{node}_j)$ in the list ConnectedList
Step  VI: Add the link to the LinksList list
Step VII: Go to Step III
```

Upon the completion of the above algorithm the selected links of the connected domain are stored in *LinksList* along with their distances. A sample entry of this list is the following:

node_i	node_j	$dist_{ij}$

Based on the entries of *LinksList* the transmission range assignments for each domain can be determined. The assignment procedure (i.e. for the heuristic optimization of 8.2) is described from the following pseudo-code:

```
for (each entry of the form $(\text{node}_i, \text{node}_j, dist_{ij})$)
{   if ( $\text{node}_i$ has not been assigned a $TxRange_i$)
          $TxRange_i = dist_{ij}$
    else {
          if ($TxRange_i < dist_{ij}$)
          $TxRange_i = dist_{ij}$
          }   }
```

Generally, in the *LinksList* among all the distances of a node to its neigboring nodes in the domain, the assignment procedure selects the maximum of them. This selection guarantees intra-domain connectivity. Whereas, in the case where the nodes select the lowest distance, it may result in two neighboring nodes selecting each other but not having connectivity to any other member of the group, resulting in partitioned domain. Furthermore, the assignment procedure results also in the establishment of bidirectional links among the members of a particular domain.

After having applied the above mechanisms, all the registered members of a domain have been assigned a transmission range, which can be translated into a specific transmission power. This transmission range (power) minimizes the average power required for intra-domain connectivity (e.g. the nodes can deliver packets to any other node of the domain) prolonging the lifespan (connectivity) of the corresponding domain.

Having ensured intra-domain connectivity, network connectivity (i.e. inter-domain connectivity) has to be established. The Border Routers (BRs) are responsible for the domains' outbound connections. The BRs are selected among the nodes of the domain, which have at least one connection to a member of neighboring domain. The BRs serve as the exits (entrances) from (to) the domain. The following paragraphs describe the selection of BRs procedure and how they are assigned transmission ranges such that the average power required for inter-domain connectivity is optimized and network connectivity is quaranteed.

Border Routers Selection Mechanism

Having defined the domains and assigned the appropriate transmission range to each node for minimum power intra-domain connectivity, inter-domain connectivity must be established along the same lines. The domains communicate each other through a set of nodes called BRs. The issue under consideration is how these BRs are being selected, such that the network is connected and the transmission power required for ensuring inter-domain connectivity is minimized.

Firstly, the selection mechanism of BRs is presented, followed by the description of the heuristics that assign the appropriate transmission range to each one of the selected BRs. The mechanism for the selection of BRs is based on the fundamental principal that for a pair of domains the nodes assigned the role of BR, are the nodes that present the minimum distance among all the pairs of nodes between the two domains. The BRs selection mechanism is represented from the following pseudo code:

```
for $i$=1,…,$K$-1 ($K$: number of domains)
    for $j$= i+1, …,$K$
    determine the $node_{ij}$ and the $node_{ji}$
    among all the nodes of domain $i$ and
    domain $j$ respectively, such that
        $min(dist(node_{ij} – node_{ji}))$
    end
end
```

The above procedure determines the set of candidate BRs:

$$\mathbb{B} = \left\{\mathbb{B}_{C(1)}, \mathbb{B}_{C(2)}, \cdots, \mathbb{B}_{C(K-1)}, \mathbb{B}_{C(K)}\right\}$$

where,

$$\mathbb{B}_{C(i)} = \left\{node_{ij}\text{: } \min\left\|node_{ij} - node_{ji}\right\|, \forall j \neq i\right\}$$

The objective is the establishment of the optimal set of candidate BRs, which will be a subset of set $\mathbb{B}$. The above algorithm determines two BRs per every possible pair of domains, so this density of BRs has to be optimized (minimized) appropriately for the establishment of the inter-domain connectivity (i.e. there is always a path from every domain i to every other domain j ($i \neq j$)) subject to the minimization of the average transmission power utilization. The description of the heuristics responsible for the selection of the set of BRs among the candidate ones and the assignment of the inter-domain transmission ranges is provided in the subsequent section.

Inter-Domain Transmission Range Assignments

Since the candidate set $\mathbb{B}$ of BRs has been determined, the assignment of the appropriate transmission ranges to a subset of them is required for minimizing the average transmission power for inter-domain connectivity. By treating $\mathbb{B}$ as a domain, the mechanism applied for the assignment of transmission ranges to the nodes of a domain can be reused without any major modification. The only twist on the functionality of the mechanism is required on the termination condition. For intra-domain connectivity the procedure terminates when all nodes in the domain get connected. Whereas, the same mechanism for establishing inter-domain connectivity (over the BRs), has to terminate when all the domains get connected.

The application of the set of mechanisms presented above results in a connected hierarchical structured network, where the average transmission power assigned to the participating nodes has been minimized. The latter is being validated in the next section, where the corresponding simulation results related to the mechanism proposed are being analyzed and evaluated.

8.2.3 Performance Evaluation

This section provides the observations and conclusions obtained from the simulation analysis of the transmission range control mechanism introduced earlier.

Specifically, indicative results for the average transmission power savings are presented in the form of shorter transmission ranges assignments for ensuring network connectivity. The proposed mechanism is being compared to the scenario where all the nodes utilize the same and the shorted possible transmission range so that the network is connected. The indicative resulting network topologies for both scenarios (proposed mechanism, common transmission range) are being provided in figures 8.1 and 8.2, respectively. These figures represent the results collected from experiments on networks of 100 nodes, which were uniformly distributed in an area of (1000m x 1000m) and were configured in 6 domains.

The performance benefits of the proposed mechanism can be indicated by visually comparing the resulting topologies of figures 8.1 and 8.2. The power utilization superiority of the proposed mechanism can be justified from the sparser resulting topology compared to the common transmission range approach. The sparsity of the links indicates that network connectivity can be achieved with smaller number of links. The proposed mechanism eliminates many of the unnecessary links as it can be observed by comparing figures 8.1 and 8.2.

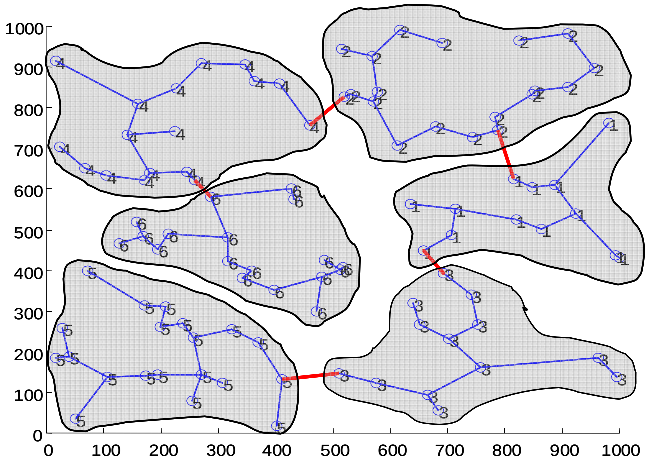


Figure 8.1. Network Topology by applying clustering and transmission range control

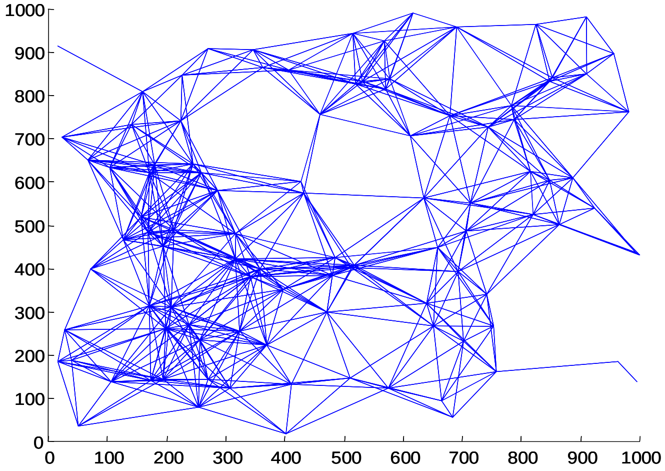


Figure 8.2. Resulting network topology by applying common transmission range

The justification of the superiority of the proposed transmission range control mechanism with respect to the average required power for ensuring network connectivity is provided from the results of figure 8.3. The results represented from this figure are related to the per node transmission range assigned and average transmission range required from both approaches, such that network connectivity is guaranteed. Similarly to figures 8.1 and 8.2, the results of figure 8.3 correspond to a network of 100 nodes, which are uniformly distributed and organized in 6 domains in an area of (1000m x 1000m).

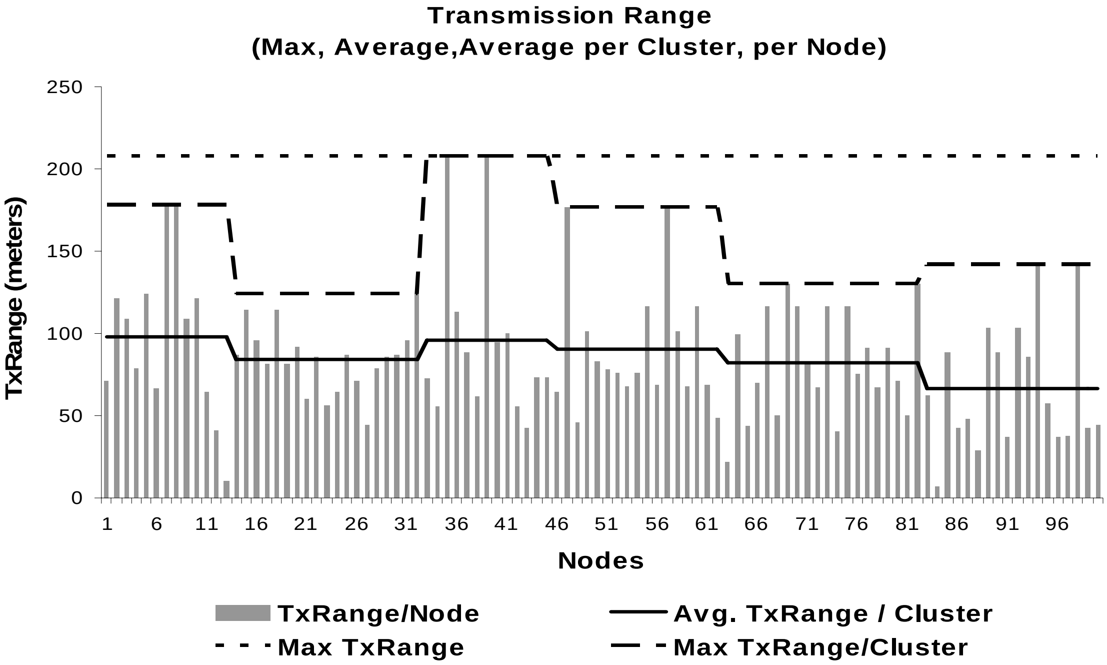


Figure 8.3. Transmission Range (max per cluster vs. avg. per cluster vs. node per cluster)

The information represented from figure 8.3, is related to the performance of both the proposed mechanism and the common transmission range approach. Specifically, figure 8.3 exploits the per node transmission range assigned from the proposed mechanism, the average and maximum transmission range per generated domain and the shorted common transmission range required from the nodes for ensuring network connectivity. The superiority of the proposed approach over the assignment of a common transmission range is justified by comparing the average transmission range assigned with the maximum one. The lower is the transmission power of the nodes, the better is the energy conservation and the network survivability is improved considerably. Furthermore, the mechanism proposed benefits also the scalability of the network, since the generation of domains creates an inherent hierarchical structure. Equally advantageous for the network performance is the lower density of the communications links, since fewer nodes will compete for accessing the shared media resulting in better MAC and end-to-end (delay, robustness) performance, due to the decrease on the number of collisions, retransmissions and packet drops.

8.3 Using Multi-objective Domain Optimization for Routing in Hierarchical Networks

Network hierarchy makes network protocols more scalable and robust, but also makes the network more complex and reduces performance. With routing protocols, hierarchy reduces overhead, routing table size, and convergence time, but can also cause sub-optimality (stretch) of the routing path length compared to the flat networks. With OSPF (Open Shortest Path First), for example, adding routing areas, with route aggregation done in Area Border Routers, can significantly reduce link state advertisements, link state database and convergence time, but can also increase inter-area path length. Existing research has produced many excellent intra- and inter-domain routing schemes and different proposals for aggregation at border routers (BRs). As important, however, is the design of the routing hierarchy itself. This section presents quantified comparisons of different hierarchical construction techniques for a simple hierarchical routing protocol in a 100 node network. When the hierarchy does not take into account routing path length suboptimality, we show the potential for significant stretch (e.g., on average more than doubling the shortest path for nodes under 6 hops apart). When we use multi-objective optimization, that includes the goal of minimizing stretch, the stretch is significantly reduced (e.g., reducing the stretch by approximately 50% for the above example). In large or dynamic environments, the introduced SA-based framework is able to produce optimized hierarchies quickly by trading a small loss in optimality for a large reduction in optimization time. The paper analyzes the choice of multi-objective cost function in this environment and concludes that simpler functions produce the best overall results.

Most networking protocols typically only work well up to some limited size, whether size is measured in terms of the number of nodes, diameter (hops) or node density (number of neighbors). As networks get larger, scaling protocols presents challenges in both the relatively stable wired internet and, more recently, in the more dynamic wireless ad hoc networks. This is clearly a challenge for routing protocols, where without careful design routing performance can degrade significantly, as measured by: a) signaling overhead (e.g., bandwidth to send routing updates), b) convergence delay (e.g., to heal failed links), c) forwarding delay (e.g., due to large table size), and d) robustness (e.g., protection from misbehaving routers). There have been significant improvements in the scalability of flat routing protocols. Particularly, Fisheye State [48] and Hazy Sighted Link State (HSLS) [49] routing scale better than most flat routing protocols. Limiting link state updates by space (hop limit) or time (e.g., updates corresponding to far-away destinations are included with lower frequency than those corresponding to near-by destinations) improves scalability. Even these protocols, however, only scale so far before overhead or performance becomes prohibitive.

Routing scalability has been a widely studied problem since the pioneering work by Kleinrock and Kamoun [46]. In their landmark paper [46], they discuss the problems associated with scalability and the length of the route forwarding table in large packet switched networks. They conclude that hierarchical routing is a requirement for very large networks.

In general, hierarchical routing groups routers into routing domains (also called clusters, Autonomous Systems, or areas) and one or more border routers (or landmarks) from each domain represent their entire domain to those outside. This process can be repeated iteratively with domains grouped into meta-domains represented by second level border routers.

Creating an efficient hierarchical structure reduces the amount of information stored, processed, and distributed. This reduction results in bandwidth saving, faster healing of faults, faster table lookups, and greater robustness. For example, domains reduce overhead by allowing aggregation of externally advertised information (e.g., using an address/label prefix to identify all nodes in a domain/cluster [47]). As important, especially in dynamic networks, changes within a domain need only be propagated within the domain. Any routing algorithm that requires routers to know about every single destination becomes infeasible as the network grows, since the size of routing tables and traffic received by each node increases in direct proportion to the number of routers.

The main disadvantage of hierarchy in routing is the sub-optimality of the path compared to shortest path routing. By isolating groups of nodes and aggregating control information, hierarchy causes routing to take an abstracted view of the network and not achieve optimal routing.

There has been a lot of design and analysis of hierarchical routing protocols and information aggregation in border routers (BRs) to provide the smallest overhead and routing table (compactness) for a given route suboptimality (stretch). As important, however, is the design of the routing hierarchy itself, which is the focus of this section. Choices include, for example, selecting how many levels of hierarchy, which nodes go in which domains, and what information to propagate at BRs.

In most networks today, the hierarchy is manually generated and is typically more a function of administrative boundaries than performance optimization. In the Internet, for example, a hierarchy is manually created using an inter-domain routing protocol (typically BGP [50]) and an intra-domain routing protocol (typically OSPF [51]). Indeed, often, intra-domain routing protocols such as OSPF are manually divided into hierarchical routing areas. In dynamic networks, hierarchy is also increasingly being used, but is generally created automatically (e.g., [39][40][52]) and there is little analysis comparing different forms of hierarchy.

This section presents quantified comparisons of different approaches to hierarchy construction and their effect on stretch. Initially, an overview of existing hierarchical routing work is given. Then the cost functions and schemes introduced for the minimization of the hierarchical routing stretch are presented, followed by their simulation analysis and evaluation.

8.3.1 Hierarchical Routing Protocols

The following subsections, initially overview the widely deployed OSPF hierarchical routing protocol, then discuss some integrated routing and distributed domain formation protocols (the theoretical bounded TZ and several hierarchical ad hoc protocols). Finally, centralized domain formation (which can be combined with hierarchal routing protocols like OSPF) is presented.

8.3.1.1 OSPF Areas

Open Shortest Path First (OSPF) is a link-state routing protocol for a single Autonomous System (AS) [51]. Each router distributes its local state by flooding the AS with Link State Advertisements. Flooding allows all routers to maintain identical link-state databases describing the current AS topology. From its topology database, each router generates its routing table by calculating a tree of shortest paths. After the internal tree is created the external routing information is examined. This external routing information may originate from another routing protocol such as BGP, or be statically configured (static routes). External routing information is flooded unaltered throughout the AS.

OSPF areas provide a two level tree, with only the special Area 0 being responsible for distributing routing information between non-backbone areas. Although all areas must be contiguous, the backbone connectivity can be established and maintained through the configuration of virtual links (through a non-backbone area).

There are three main types of OSPF routers: Internal, Area Border, and AS boundary. Internal routers run a single copy of the basic routing algorithm. The Area border routers run multiple copies of the basic OSPF link-state algorithm (and separate link-state databases) for each area it is connected to. Area border routers condense the topological information a) of their attached areas for distribution to other areas, b) its cost to all networks external to the area to its internal routers. AS boundary routers advertises external routing information throughout the AS. The paths to each AS boundary router are known by every router in the AS.

The topology of an area is typically invisible outside of the area. This isolation of knowledge enables the protocol to greatly reduce routing traffic as compared to treating the entire AS as a single link-state domain. Also, routing within the area (intra-area routing) is determined only by the area's own topology, lending the area protection from bad routing data. There are different kinds of non-backbone areas depending on the amount of aggregation at border routers [56]. This becomes a key choice in balancing routing overhead, table size, convergence time and routing sub-optimality [57].

8.3.1.2 Thorup-Zwick (TZ) routing hierarchy

The Thorup-Zwick (TZ) routing scheme [58] provably has worst routing path of no more than 3 times the optimal (flat) routing scheme (stretch-3) yet delivers a nearly optimal routing table size. TZ begins (step 1) by interactively selecting Landmarks [47] from the set of nodes. The selection itself is done in several rounds, with available nodes selected using a uniform random probability. At the end of each round, nodes join to their closest landmark. These Landmark selection rounds end when all clusters are below a certain size and we have a network with a two-level hierarchy. The essence of the TZ scheme is the right balance between the number of landmarks and the cluster sizes. In Step 2 every node calculates its outgoing port for the shortest path to every Landmark and every node in its cluster. Step 3 configures the node labels to reduce the table size (like CIDR address configuration).

Although this protocol is not practical in large dynamic networks, the importance of this work is that it shows that through the introduction of hierarchy in large networks and address/label configuration according to that hierarchy, it is possible to tradeoff a small loss in route optimality to significantly improve the scalability of networking protocols. Indeed, Krioukov, Fall and Yang [59] have shown that the stretch is actually much less than 3 for scale-free networks such as the Internet (i.e., where any of the 10,000 backbone nodes is at most approximately 6 hops from every other node). They show 70% of the routes are optimal and the average stretch is only around 1.1 (10% more than the shortest path).

8.3.1.3 Hierarchical ad hoc routing Protocols

There is a large diversity in the ad hoc routing protocols [60] [61], but amongst the most scalable are three that create a two level hierarchy. The Zone Routing Protocol (ZRP) [19] proactively maintaining routes to destinations within a local neighborhood (lowest level of the hierarchy). Reactive routing is used to determine routes to destinations outside the zone using peripheral nodes. Landmark Ad Hoc Routing (LANMAR [62] [63]) uses a combined link state and distance vector routing protocol using the Landmark idea [47], but without requiring predefined hierarchical addresses. The Optimized Link State Routing (OLSR) Protocol [64] uses multi-point relays to reduce the number and size of link state update messages. Each node determines a subset of its neighbors as multipoint relay nodes to propagate its link state updates.

8.3.1.4 Global hierarchy formation protocols

The global approach to creating a hierarchy is a computationally-intensive centralized graph partitioning procedure using information from all network elements. While the graph partitioning problems is known to be *NP*-Complete, many heuristic solutions have been investigated including Kernighan-Lin [65] and Simulated Annealing [35] heuristics. Application of graph partitioning to create an efficient hierarchical structure began with the pioneering work by Steenstrup, Ramanathan, and Krishnan at BBN [52] [40]. They focused on the use of Kernighan-Lin to create a hierarchy for a single link state routing protocol. This work has been generalized to include any network functions and allow the use of different routing protocols [39]. Real time optimization has been shown even for large networks using Simulated Annealing [10] [54]. Finally, the complementary combination of global and local optimization to create hierarchical routing domains is described in [55].

8.3.2 Minimizing the hierarchical routing path length suboptimality

It is known that the application of hierarchy introduces routing path length suboptimality. The objective is to propose solutions, independent of the hierarchical routing scheme applied, which will be able to minimize the suboptimality. The schemes introduced utilize the existing SA-based framework. Due to the utilization of the SA-based framework the difficulty of the design is mainly located on the introduction of the appropriate cost functions that consider the hierarchical routing path length (HRPL) suboptimality and attempt to minimize it. One of the novelties of the proposed approach is that the minimization of the routing path length suboptimality is attempted in conjunction with meeting also other hierarchy generation objectives, unrelated to the routing suboptimality. Such an approach, attempts to exploit the multi-objective optimization capabilities of SA for the generation of hierarchies able to satisfy multiple performance objectives, simultaneously.

Depending on the problem assumptions three different schemes are adopted for minimizing the HRPL suboptimality:

- **Scheme 1:** Given the hierarchy the domains' clusterheads (CHs) must be selected subject to the minimization of HRPL suboptimality.
- **Scheme 2:** Given the set of CHs the hierarchical structure must be generated so that both the routing path length suboptimality is minimized and the hierarchy generation objectives are satisfied.
- **Scheme 3:** This scheme combines schemes 1 and 2. The objective is the combined selection of CHs and generation of hierarchy so that both the HRPL suboptimality is minimized and other hierarchy generation objectives are satisfied.

In the context of hierarchical routing, the CH as an entity is defined to be a special node that represents its domain in the hierarchical routing functions. An example of such an entity is provided in section 3.3.1.

8.3.2.1 Scheme 1: Selecting the CHs on a given hierarchical structure

The assumption of this scenario is that the hierarchy has already been generated with respect to a set of pre-specified objectives that are not related to the routing path length suboptimality. Such objectives, as they have been presented in [53], could be the generation of balanced size or balanced diameter clusters, the minimization of border routers, the grouping of nodes with similar mobility characteristics, or combinations of them. Given the hierarchical structure, a set S of special nodes (CHs) has to be selected, so that the HRPL suboptimality is minimized with respect to the applied hierarchical routing scheme R. The proposed cost function that represents the latter objective is:

$$J\left(S\right)=\min_{S}\sum_{i=1}^{\left|F_{SD}\right|}\left(HRPL_{i,R}-FRPL_{i}\right) \quad \text{subject to } C^{*}\wedge F_{SD}\wedge R \tag{8.3}$$

where,

F_{SD}: Set of source and destination pairs

R: Hierarchical routing scheme applied

$HRPL_{i,R}$: Hierarchical routing path length (hops) for the *i_{th}* *S-D* pair when R is applied

$FRPL_{i}$: Flat routing path length (Dijkstra's hops) for the i_{th} *S-D* pair

C^{*}: Optimal Clustering map with respect to a set of hierarchy generation objectives

S: Set of CHs

The optimization of (8.3) from the SA-based framework will provide the optimal set of CHs S^{*}, which will minimize the HRPL suboptimality without affecting the hierarchical structure and the generation objectives it satisfies.

8.3.2.2 Scheme 2: Generating the hierarchical structure given the set S of CHs

As opposed to the previous scenario, in this scenario it is assumed that that set S of CHs is provided. The objective is to generate the hierarchical structure C^{*}, which will satisfy a set of hierarchy generation objectives. These objectives can be independent of the HRPL suboptimality. The hierarchy generation process has to be aware of the HRPL suboptimality which will attempt to minimize given the set of CHs S and a hierarchical routing scheme R. This scenario belongs to the multi-objective optimization class of problems, since the hierarchy generation objectives and the minimization of HRPL suboptimality is preferred to be satisfied simultaneously. Given that the cost function representing the non-HRPL hierarchy generation objectives is $J_H\left(C\right)$ then the cost function to be optimized for providing a solution to the multi-objective problem is:

$$J\left(C\right)=\min_{C}\left[J_H\left(C\right)+10^2 * \sum_{i=1}^{\left|F_{SD}\right|}\left(HRPL_{i,R}-FRPL_i\right)\right] \tag{8.4}$$
$$\text{subject to } S \wedge F_{SD} \wedge R$$

The optimization of (8.4) from the SA-based framework will provide that the hierarchical structure capable of simultaneously satisfying the set of hierarchy generation objectives and minimize HRPL suboptimality.

8.3.2.3 Scheme 3: Combined HRPL minimization and hierarchy generation

This scenario combines the two schemes described in sections 3.2.1 and 3.2.2. This scenario, as the scenario described in 3.2.2, can also be considered as a multi-objective optimization problem. The difference with the scenario described in 3.2.2 is that the set S is not provided beforehand but needs to be optimally specified dynamically. The cost function that represents this scenario is:

$$J\left(C,S\right)=\min_{C,S}\left[J_H\left(C\right)+10^2 * \sum_{i=1}^{\left|F_{SD}\right|}\left(HRPL_{i,R}-FRPL_i\right)\right] \tag{8.5}$$
$$\text{subject to } F_{SD} \wedge R$$

This scenario is the most complicated of the three since S and C^* have to be constructed simultaneously.

8.3.3 Performance Evaluation of the HRPL minimization schemes

Even though the schemes introduced are independent of the implied routing scheme R and the set of hierarchy generation objectives, in order to be evaluated a “generic” hierarchical routing protocol and a set of hierarchy generation objectives have to be assumed. Prior to the simulation analysis and evaluation of the proposed schemes, the assumptions of the experiments performed are presented. Initially, the HRPL suboptimality with respect to R is being quantified. Then the evaluation of the proposed schemes is done with respect to their ability on improving the HRPL suboptimality.

8.3.3.1 Generic hierarchical routing protocol

There are many different ways to route packets hierarchically. For the evaluation of the proposed schemes a simple generic hierarchical routing protocol is assumed. The behavior of the protocol differs depending on whether the source (*S*) and destination (*D*) belong (or not) in the same domain. In the case where the *S*-*D* pair belongs in the same domain, the protocol functions as an optimal flat routing protocol and the *S*-*D* distance (hops) is defined with respect to the *S*-*D* Dijkstra distance. When source *S* and destination *D* belong in different domains, the hierarchical routing functionality is represented from figure 8.4.

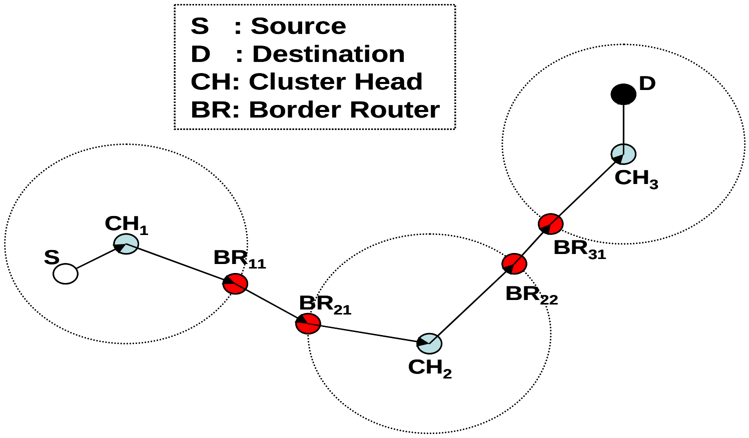


Figure 8.4. Simple Hierarchical Routing Scheme

The important entities in this hierarchical routing scheme are the **CHs** and the border routers (**BRs**). The CHs abstract the domain in the inter-domain routing updates, and the BRs are nodes that have links to domains other than their own. The hierarchical routing is performed by having the source *S* send to its Cluster Head (**CH**), which in turn sends to a neighboring CH towards the domain of the destination *D*, through the corresponding Border Routers (**BRs**).

The hierarchical route path length $HRPL\left(S_k^i, D_k^j\right)$ for the *k*th source destination *S-D* pair, where the source S_k^i belongs in domain i and the destination D_k^j belongs in domain $j\ \left(j \neq i\right)$ is defined as:

$$HRPL\left(S_k^i, D_k^j\right) = d\left(S_k^i, CH^i\right) + d\left(CH^i, CH^j\right) + d\left(CH^j, D_k^j\right) \qquad (8.6)$$

where,

$d\left(S_k^i, CH^i\right)$: shortest intracluster distance from source S_k^i node to its CH^i

$d\left(CH^i, CH^j\right)$: shortest intercluster distance from the CH^i of the source node to the CH^j of the destination node, through neighboring CHs and BRs

$d\left(CH^j, D_k^j\right)$: shortest intracluster distance from destination's CH^j to the destination D_k^j.

The intra-domain CH-BRs distances are defined by Dijkstra’s algorithm. Inter-domain distances among CHs are defined by abstracting the network using only the CHs and BRs and then applying Dijkstra’s algorithm.

8.3.3.2 Hierarchy Generation Objectives

For the performance evaluation purposes a hierarchy generation objective has been selected from [53], which aims on the generation of balanced size clusters. The representative cost function is:

$$J_H\left(C\right) = \min_{C} Var\left(\left|C_1\right|^2, \ldots, \left|C_K\right|^2\right) \tag{8.7}$$

where,

$\left|C_i\right|$: Cardinality (size) of i^{th} cluster

K: Number of generated clusters

8.3.3.3 Quantifying HRPL suboptimality

Prior to the evaluation of the proposed schemes some representative results related to the HRPL suboptimality will be provided, when the hierarchy generation process does not take it into account. For this purpose, the generic hierarchical routing protocol described above was applied on a hierarchical structure generated with respect to the balanced size clusters objective. In each generated domain the node with the lowest ID (LID) is selected as the CH. The following results (average over 100 experiments) quantify the size of the suboptimality for different optimal path lengths (Flat Routing Path Length - FRPL) in a network of 100 nodes where 4 domains were generated (figure 8.5).

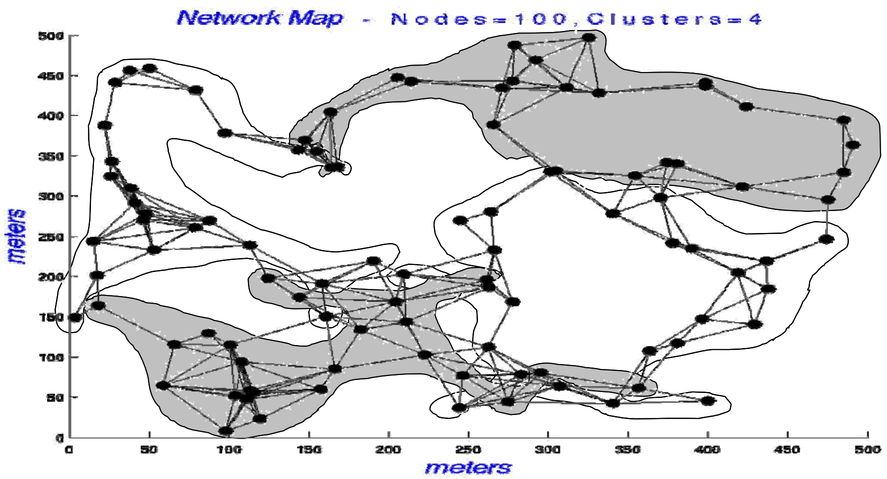


Figure 8.5. Simulated Network Topology

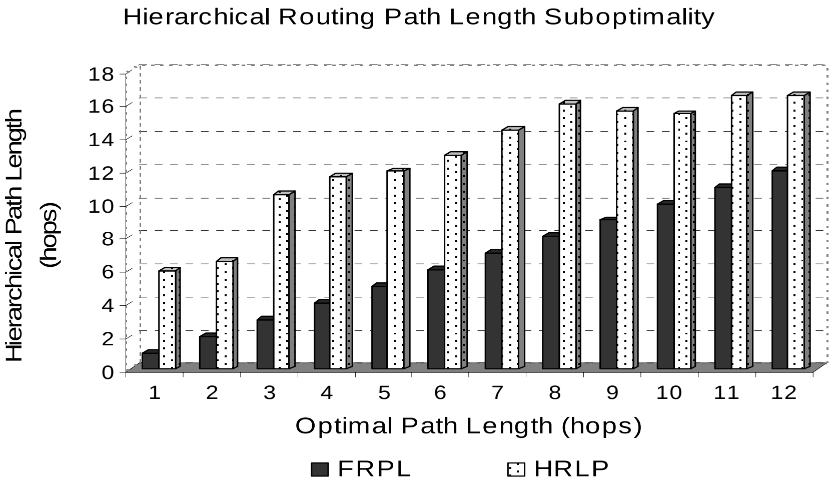
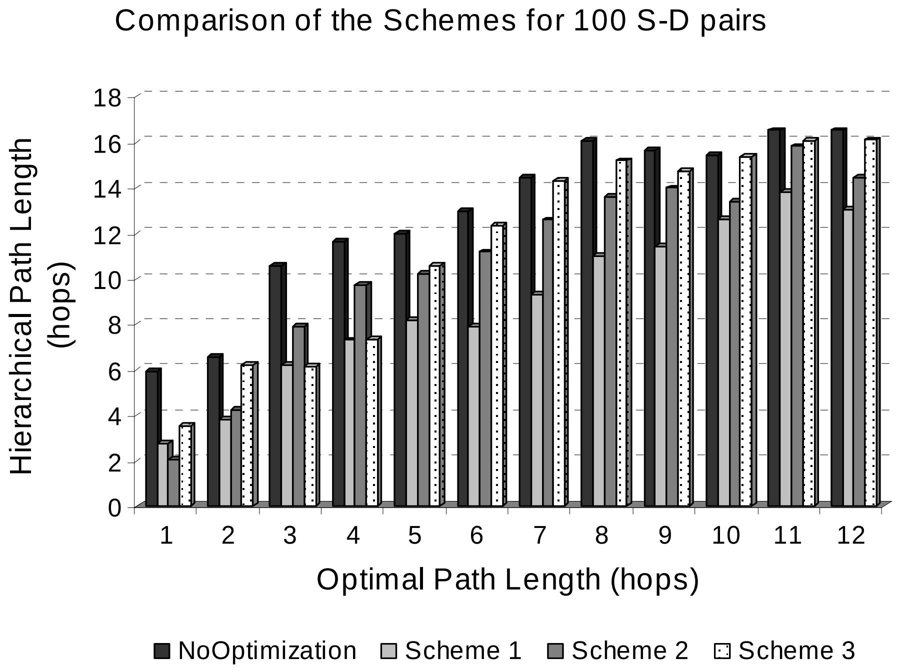


Figure 8.6. Quantification of suboptimality

Figure 8.6 is representative of the effect of hierarchy on routing when it is not taken into account when this hierarchy is being generated. There are cases where the resulting hierarchical path is more than double of the optimal path length. This can significantly degrade the QoS performance of the network, since the packets will have to traverse more links causing more collisions on the MAC layer. Due to this side effect, when $HRPL = K * FRPL$, $K \in \mathbb{Z},\ K > 1$, worse degradation may be introduced to the observed end-to-end delay T_{end} $(T_{end}^{HRPL} >> K * T_{end}^{FRPL})$.

8.3.3.4 Evaluation of the schemes

The schemes proposed earlier have the ability to improve the HRPL suboptimality and at the same time to construct hierarchical structures that improve various other aspects of the network performance. For performance evaluation purposes an indicative non-HRPL hierarchy generation objective (generation of balanced size clusters) is enforced along with the minimization of hierarchical path length suboptimality. The simulation results demonstrated on figure 8.7 reveal the degree of effectiveness of each of the proposed schemes. These results correspond to a network of 100 nodes where 4 domains have been generated. The upper graph of figure 8.7 corresponds to the coexistence of 100 *S-D* pairs into the network and the lower graph corresponds to the coexistence of 25 *S-D* pairs, respectively.



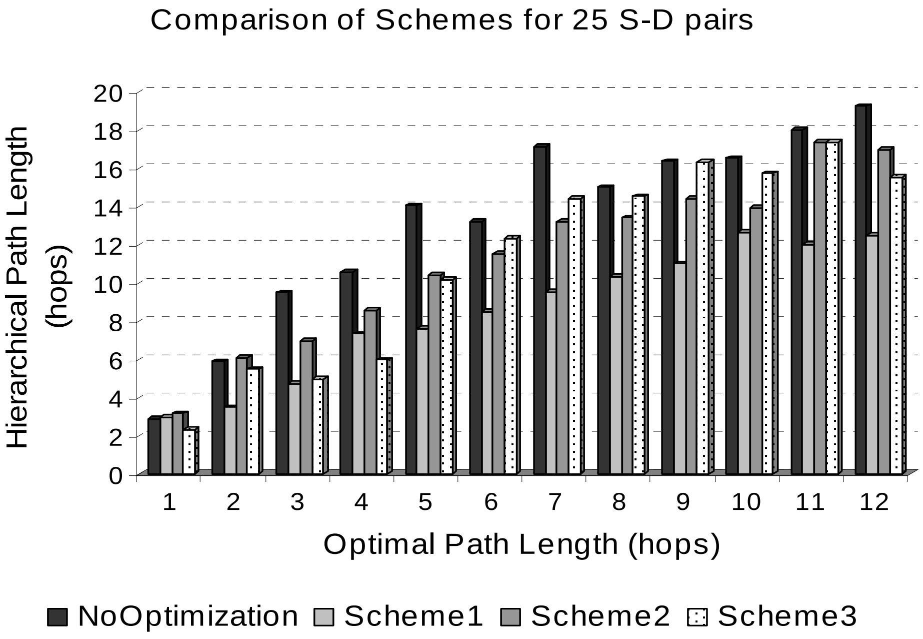


Figure 8.7. Comparison of the proposed schemes

The main observation in terms of the schemes is that all of them improve the suboptimality. Scheme 1 (selection of CHs given the hierarchical structure) consistently outperforms the others by producing structures that provide paths that are at most 3 hops from the optimal, independently of the optimal path length. This can be explained due to the simpler form of its corresponding cost function (8.3) compared to the more complex (8.4) and (8.5). The SA algorithm that constitutes the cornerstone of the developed SA-based framework has been modified and adjusted for speed trading off its optimality. So, for complex cost functions with large solution spaces, the solutions obtained are more likely to be less optimal compared to the solutions obtained for simpler with smaller solutions spaces cost functions. Even though Scheme 3 attempts to solve a very complicated problem (simultaneously select the CHs and generate the hierarchy); manages to improve the HRPL suboptimality and satify the non-HRPL objectives. Furthermore, it does this best when fewer *S-D* pairs coexist into the network, because of the smaller solution space of the optimization problem. Generally all 3 schemes perform better when less *S-D* pairs coexist (i.e. smaller solution spaces of the corresponding optimization problems). The performance (e.g. minimization of the HRPL stretch) difference among the three schemes proposed; becomes smaller because the optimization problem complexity is reduced, so the faster SA-based framework can converge to better solutions.

Bibliography

[1] H. W. Simpson, “*The Point to Point Protocol (PPP)*,” Internet STD 51, July 1994.

[2] W. Townsley, et al, “*Layer Two Tunneling Protocol ‘L2TP’*,” RFC 2661, August 1999.

[3] L. Mamakos, et al., “*Method for Transmitting PPP Over Ethernet (PPPoE)*,” RFC 2561, February 1999.

[4] R. Droms, “*Dynamic Host Configuration Protocol*,” RFC 2131, March 1997.

[5] E. Guttman, “*Autoconfiguration for IP Networking: Enabling Local Communication*,” IEEE Internet Computing, Vol. 5, Issue 3, pp. 81 – 86, May 2001.

[6] H. Zhou, L. Ni, and M. Mutka, "*Prophet Address Allocation for Large Scale MANETs*," in Proceedings of INFOCOM 2003, San Francisco, CA, April 2003.

[7] Charles E. Perkins, Jari T. Malinen, Ryuji Wakikawa, Elizabeth M. Belding-Royer, Yuan Sun, “*IP Address Autoconfiguration for Ad Hoc Networks*,” draft-ietf-manetautoconf-01.txt, November 2001.

[8] S. Thomson and T. Narten, “*IPv6 stateless address autoconfiguration*,” RFC 2462, Internet Engineering Task Force, December 1998.

[9] N. Vaidya, "*Weak duplicate address detection in mobile ad hoc networks*," in Proc. ACM MobiHoc'02, 2002.

[10] S. Nesargi, R. Prakash, "*MANETconf: configuration of hosts in a mobile ad hoc network*," Joint Conference of the IEEE Computer and Communications Societies, INFOCOM `02, pp. 1059-1068, June 2002.

[11] K. Weniger, "*PACMAN: Passive Autoconfiguration for Mobile Ad hoc Networks*," IEEE Journal on Selected Areas in Communications (JSAC), Special Issue 'Wireless Ad hoc Networks', March 2005.

[12] M. Mohsin, R. Prakash, "*IP Address Assignment in A Mobile Ad Hoc Network*," MILCOM 2002, pp. 856–861, Anaheim, CA, October 2002.

[13] A. McAuley, A. Misra, L. Wong, K. Manousakis, "*Experience with Autoconfiguring a Network with IP Addresses*," Proceedings of IEEE MILCOM, October 2001, Fairfax, USA.

[14] P. R. Wilson, M. S. Johnstone, M. Neely, and D. Boles, "*Dynamic Storage Allocation: A Survey and Critical Review*," In H.G. Baker, editor, Proceedings of the International Workshop on Memory Management (IWMM'95), Kinross, Scotland, UK, volume 986 of Lecture Notes in Computer Science, pp. 1-116, Springer-Verlag, Berlin, Germany, 1995.

[15] A. J. McAuley and K. Manousakis, "*Self-Configuring Networks*," IEEE MILCOM 2000 Conference Proceedings, Los Angeles, CA, October 2000.

[16] A. McAuley, Subir Das, S. Madhani, S. Baba and Y. Shobatake, "*Dynamic Registration and Configuration Protocol*," IETF Draft, draft-itsumo-drcp-01.txt., 2001

[17] R. Lin and M. Gerla, “*Adaptive Clustering for Mobile Wireless Networks,*” IEEE Journal on Selected Areas in Communications, pages 1265-1275, September 1997

[18] M. Gerla and J. T. Tsai, “*Multiuser, Mobile, Multimedia Radio Network*,” Wireless Networks, vol. 1, Oct. 1995, pp. 255–65.

[19] D. Baker, A. Ephremides, and J. Flynn “*The design and simulation of a mobile radio network with distributed control*,” IEEE Journal on Selected Areas in Communications, SAC-2(1):226--237, 1984

[20] A. Ephremides, J. E. Wieselthier, and D. J. Baker, “*A Design Concept for Reliable Mobile Radio Networks with Frequency Hopping Signaling*,” in Proc. IEEE, vol. 75, 1987, pp. 56–73.

[21] M. Chatterjee, S. K. Das, D. Turgut, “*WCA: A Weighted Clustering Algorithm for Mobile Ad hoc Networks*, “ Journal of Cluster Computing (Special Issue on Mobile Ad hoc Networks), Vol. 5, No. 2, April 2002, pp. 193-204

[22] S. Basagni, “*Distributed and Mobility-Adaptive Clustering for Multimedia Support in Multi-Hop Wireless Networks*,” Proceedings of Vehicular Technology Conference, VTC 1999-Fall, Vol. 2, pp. 889-893

[23] J. Wu and H. L. Li, “*On Calculating Connected Dominating Set for Efficient Routing in Ad Hoc Wireless Networks*,” Proc. 3rd Int’l. Wksp. Discrete Algorithms and Methods for Mobile Comp. and Commun., 1999, pp. 7–14.

[24] Y.-Z. P. Chen and A. L. Liestman, “*Approximating Minimum Size Weakly-Connected Dominating Sets for Clustering Mobile Ad Hoc Networks*,” in Proc. 3rd ACM Int’l. Symp. Mobile Ad Hoc Net. & Comp., June 2002, pp. 165–72.

[25] C.-C. Chiang et al., “*Routing in Clustered Multihop, Mobile Wireless Networks with Fading Channel,*” in Proc. IEEE SICON’97, 1997.

[26] J. Y. Yu and P. H. J. Chong, *“3hBAC (3-hop between Adjacent Clusterheads): a Novel Non-overlapping Clustering Algorithm for Mobile Ad Hoc Networks,*” in Proc. IEEE Pacrim’03, vol. 1, Aug. 2003, pp. 318–21.

[27] T. J. Kwon et al., “*Efficient Flooding with Passive Clustering — an Overhead-Free Selective Forward Mechanism for Ad Hoc/Sensor Networks*,” in Proc. IEEE, vol. 91, no. 8, Aug. 2003, pp. 1210–20.

[28] P. Basu, N. Khan, and T. D. C. Little, “*A Mobility Based Metric for Clustering in Mobile Ad Hoc Networks,*” in Proc. IEEE ICDCSW’01, Apr. 2001, pp. 413–18.

[29] A. B. McDonald and T. F. Znati, “*Design and Performance of a Distributed Dynamic Clustering Algorithm for Ad-Hoc Networks*,” in Proc. 34th Annual Simulation Symp., Apr. 2001, pp. 27–35.

[30] A. D. Amis and R. Prakash, “*Load-Balancing Clusters in Wireless Ad Hoc Networks,*” in Proc. 3rd IEEE ASSET’00, Mar. 2000, pp. 25–32.

[31] J. Wu et al., “*On Calculating Power-Aware Connected Dominating Sets for Efficient Routing in Ad Hoc Wireless Networks,*” J. Commun. and Networks, vol. 4, no. 1, Mar. 2002, pp. 59–70.

[32] J.-H. Ryu, S. Song, and D.-H. Cho, “*New Clustering Schemes for Energy Conservation in Two-Tiered Mobile Ad-Hoc Networks*,” in Proc. IEEE ICC’01, vo1. 3, June 2001, pp. 862–66.

[33] T. Ohta, S. Inoue, and Y. Kakuda, “*An Adaptive Multihop Clustering Scheme for Highly Mobile Ad Hoc Networks,*” in Proc. 6th ISADS’03, Apr. 2003.

[34] C.H. Papadimitriou, K. Steiglitz, "*Combinatorial Optimization: Algorithms and Complexity*," Prentice Hall, New York, 1982.

[35] S. Kirkpatrick, C. D. Gelatt, Jr., and M. P. Vecchi, "*Optimization by Simulated Annealing*," Science 220 (13 May 1983), 671-680.

[36] V. Cerny, "*Thermodynamical Approach to the traveling salesman problem: An efficient simulation algorithm*," Journal of Optimization Theory and Applications 45, 1985, pp. 41-51.

[37] B.W. Kernighan, S. Lin, "*An efficient heuristic procedure for partitioning graphs*," Bell System Technical Journal 49, 1970, pp. 291-307.

[38] R. Morera, A. McAuley, "*Flexible Domain Configuration for More Scalable, Efficient and Robust Networks*," MILCOM, October 2002.

[39]. K. Manousakis, J. McAuley, R. Morera, J. Baras, "*Routing Domain Autoconfiguration for More Efficient and Rapidly Deployable Mobile Networks*," Army Science Conference 2002, Orlando, FL

[40]. Rajesh Krishnan, Ram Ramanathan and Martha Steenstrup, "*Optimization Algorithms for Large Self-Structuring Networks*," Proceeding of IEEE INFOCOM '99, New York, USA, March 1999.

[41] T.A. ElBatt, S.V. Krishnamurthy, D. Connors, S. Dao, "*Power Management for Throughput Enhancement in Wireless Ad-Hoc networks*," IEEE ICC'00, New Orleans, LA, June 2000

[42] Narayanaswamy S., Kawadia V., Sreenivas R.S, Kumar P.R, "*Power control in ad-hoc networks: Theory, architecture, algorithm and implementation of the COMPOW protocol*," EWC 2002

[43] R. Ramanathan, R. Rosales-Hain, "*Topology Control of Multihop Wireless Networks using Transmit Power Adjustment*," Procedings of INFOCOM 2000

[44] R. Wattenhofer, L. Li, P. Bahl, and Y.-M. Wang, "*Distributed topology control for power efficient operation in multihop wireless ad hoc networks*," in Proceedings of INFOCOM, 2001, pp. 1388—1397

[45] Kawadia V., Kumar P.R., "*Power Control and Clustering in Ad Hoc Networks*," IEEE INFOCOM, April 2003.

[46] L. Kleinrock, F. Kamoun, "*Hierarchical Routing for Large Networks: Performance Evaluation and Optimization*," Computer Networks, Vol. 1, pp. 155-174, January 1977.

[47] P. F. Tsuchiya, "*The landmark hierarchy: A new hierarchy for routing in very large networks*," Computer Communications Review, vol. 18, no. 4, pp. 43–54, August 1988.

[48] Guangyu Pei et al., "*Fisheye State Routing: A Routing Scheme for Ad Hoc Wireless Networks*," ICC 2000, June 2000.

[49] Cesar A., Santiváñez et al., "*Making Link-State Routing Scale for Ad Hoc Networks,*" MOBIHOC 2001.

[50] Y. Rekhter, T. Li, "*A Border Gateway Protocol 4 (BGP-4)*," RFC 1771, March 1995.

[51] J. T. Moy, "*OSPF: Anatomy of an Internet Routing Protocol,*" Reading, MA: Addison-Wesley, 1998.

[52] R. Ramanathan, M. Steenstrup, "*Hierarchically Organized, Multihop Mobile Networks for QoS Support*," ACM/Baltzer Mobile Networks and Applications, Vol.3, No.1, pp.101-19, January 1998.

[53] K. Manousakis, A. McAuley, R. Morera, "*Applying Simulated Annealing for Domain Generation in Ad Hoc Networks*," IEEE International Conference on Communications (ICC), Paris, June 20004.

[54] K. Manousakis, J. Baras, A. McAuley, R. Morera, "*Improving the Speed of Dynamic Cluster Formation in MANET via Simulated Annealing*," 4th Army Science Conference (ASC), Orlando, Florida, November, 2004.

[55] K. Manousakis, A. McAuley, R. Morera, J. Baras, "*Rate of Degradation of Optimization Solutions and its Application to High Performance Domain Formation in Ad Hoc Networks*," IEEE/AFCEA MILCOM 2004. Monterey. CA, October 2004.

[56] A. Zinin, A. Lindem, D. Yeung, "*Alternative Implementations of OSPF Area Border Routers*," Request for Comments: 3509, April 2003

[57] R. Rastogi, Y. Breitbart, M. Garofalakis, and A. Kumar "*Optimal Configuration for OSPF Aggregates*", IEEE/ACM Transactions on Networking, Vol. 11, No. 2, April 2003.

[58] M. Thorup and U. Zwick, "*Compact routing schemes*," In Proceedings of the 13th Annual ACM Symposium on Parallel Algorithms and Architectures (SPAA), pages 1-10, 2001.

[59] D. Krioukov, K. Fall, and X. Yang, "*Compact Routing on Internet-like Graphs*," Proceedings of IEEE INFOCOM 2004, March 2004.

[60] Elizabeth M. Royer et al., "*A Review of Current Routing Protocols for Ad Hoc Mobile Wireless Networks*," IEEE Personal Communications, April 1999.

[61] Z. Haas, M. Perlman, "*ZRP: A Hybrid Framework for Routing in Ad Hoc Networks*," In Ad Hoc Networks, Ed. Charles E. Perkins, Addison-Wesley, 2001.

[62] Kaixin Xu et al., "*Landmark Routing in Ad Hoc Networks with Mobile Backbones*," Journal of Parallel and Distributed Computing, Special Issue on Ad Hoc Networks, 2002.

[63] Mario Gerla et al., "*Landmark Routing for Large Ad Hoc Wireless Networks*," GLOBECOM 2000, November 2000.

[64] Thomas Clausen, Philippe Jacquet, "*Optimized Link State Routing Protocol*," Internet Draft, IETF MANET Working Group, draft-ietf-manet-olsr-11.txt, July 2003.

[65] S. Lin & B. W. Kernighan, "*An Effective Heuristic Algorithm for the Traveling-Salesman Problem*", Oper. Res. 21, 498-516 (1973).

[66] X. Hong, M. Gerla, G. Pei, and C. Chiang, "*A group mobility model for ad hoc wireless networks*," in Proceedingsof the ACM International Workshop on Modeling and Simulation of Wireless and Mobile Systems (MSWiM), August 1999.