Università degli studi di Pisa

Facoltà di Scienze Matematiche, Fisiche e Naturali

Corso di Laurea Specialistica in Tecnologie Informatiche

Tesi di laurea

LVMM - The Localized Vehicular Multicast Middleware: a Framework for Ad Hoc Inter-Vehicles Multicast Communications

Candidato
Filippo Barsotti

Relatore
prof. Stefano Chessa

Correlatore
prof.ssa Laura Ricci

Anno Accademico 2004-2005

Alla mia mamma

Ringraziamenti

Vorrei esprimere la mia più sincera gratitudine al relatore di questa tesi, il prof. Stefano Chessa per il suo fondamentale supporto e guida attraverso tutti i passi di questa ricerca. Ringrazio infine mia mamma, i miei amici e chiunque mi è stato vicino per il loro supporto di ogni giorno.

Acknowledgements

I would like to express my sincere gratitude to my thesis supervisor prof. Stefano Chessa for his invaluable support and guidance all through this research endeavour. I would like to thank my mum, friends and anybody close to me for their support in all respects.

Introduction

The subject of this thesis is wireless ad hoc communication in vehicular networks.
This thesis defines a novel semantic for *multicasting* so that it can be utilized in vehicular networks and it defines and describes a *middleware* that allows applications to transparently exploit such multicasting.
Inter-vehicles communications are a promising area for ad hoc networks deployment; still they present several challenges that will have to be faced in order to realize working and useful solutions for this environment.
This thesis defines and describes a middleware enabling multicast communications in vehicular ad hoc networks (VANETs): the proposed framework allows distributed low-latency multicast applications to execute in a highly mobile environment like vehicular networks.
Ideally, it is desirable to guarantee that the QoS requirements of on-going connections are preserved for their entire duration. Unfortunately, this is not possible in a time-varying network environment as connections may fail randomly due to user mobility. A more realistic and practical approach is to provide some form of probabilistic QoS guarantees by keeping connection failures below pre-specified threshold value and by ensuring, with high probability, that a minimum level of bandwidth is always available to ongoing connections.
The proposed middleware performs two main tasks: it finds vehicles suitable to support a multicast communication and it transparently routes multicast messages among those vehicles. That is, this framework finds and maintains groups of vehicles suitable to run distributed multicast applications and it executes a multicasting protocol, the *Vehicular Multicast Routing Protocol* (VMRP) that delivers messages to all the recipients; applications just have to subscribe to this middleware, without worrying about the underlying network.
VMRP is an ad-hoc solution for vehicular networks; it delivers packets to all members utilizing a loop-free, minimum cost path from each source to all the recipients. It does not require a vehicle to know all other members: only knowledge of directly reachable nodes is required to perform the source-based routing.
With such multicast routing each node maintains a view of the global membership, thus each node knows all other members of a group. This is totally transparent to applications.
The ultimate goal of LVMM is to assist software developers in their efforts to design and build mobile applications over ad-hoc networks. The key to LVMM strategy is to provide, at the application level, the appearance of stability in a domain that is characterized by high degrees of mobility.

The starting point of this thesis was the research of articles about inter-vehicles communications to find out their characteristics, their peculiarities, the actual research done on this argument and the relevant open problems.
What should have been the specific subject of this thesis was not defined: the first thing to do was understand VANETs, find out what already was done and find out an interesting argument that did not already have a published solution.
Before selecting and defining the subject of this paper, a study on ad hoc inter-vehicles communications, ad hoc routing, interesting vehicular applications and the differences between VANETs and other ad hoc networks has been lead.
Thus, this thesis begun with those studies that are subsumed in the initial chapters of this document. In particular, the focus was on routing issues and algorithms, because routing is a fundamental and still open problem and applications for ad hoc networks have a tight coupling with the services offered by the underlying layers.
Positional routing is a new approach that will be very useful for ad hoc networks, especially for VANETs.

Chapter 1 is an introduction to ad hoc networks. VANETs have their unique peculiarities but their description cannot leave out of consideration a generic analysis of ad hoc networks. Ad hoc networks' properties and routing issues are exposed in this chapter.

Chapter 2 presents the characteristics of a Vehicular Ad Hoc Network, showing its unique properties with respect to other ad hoc environments.

Chapter 3 introduces the positional routing enabled by position tracking devices by describing the GPS: the Global Positioning System.

Chapter 4 describes properties of positional routing and some existing algorithms.

Chapter 5 introduces Geocasting, a useful and interesting mechanism that might be at the base of some future communication patterns for vehicular networks.

Chapter 6 introduces Multicasting. This chapter describes the argument that has been the starting point for this thesis. It defines what multicasting means, what it does imply, which are the relevant aspects in ad hoc environments and it presents some existing algorithms.
Within this chapter there is a definition of an interesting mechanism called Geomulticasting.

Chapter 7 describes a middleware - LVMM - that represents a solution to low-latency

multicasting in vehicular networks. This is the argument developed for this thesis.

LVMM allows vehicles to exchange multicast packets; a multicast application will simply rely on LVMM to transparently send/receive multicast messages.

LVMM will find out a set of vehicles suitable to exchange multicast packets and will perform the multicast routing to deliver each packet sent by a member to all other members.

LVMM allows low-latency multicast applications with QoS requirements to execute in a vehicular network: it enables execution of applications like video-conferencing, real-time messaging, distributed games, chats, etc.

This way it also allows groups of vehicles to exchange data about road and traffic conditions in an ad hoc manner and enables applications like Cooperative Adaptive Cruise Control.

A multicast protocol for vehicular networks is proposed in this chapter. This is the protocol that LVMM utilizes to deliver messages to all the members of a group.

This protocol performs multicasting utilizing a loop-free, minimum cost path from each source to all the destinations without requiring each node to know the complete topology of a group.

Thus, the proposed protocol is simple, scalable, lightweight, decentralized and yet utilizes least cost paths to deliver every packet sent by each source of multicast traffic to all the destinations.

Table of Contents

Chapter 1

Introduction to Ad Hoc networks

In the near future, a pervasive computing environment can be expected based on the recent progresses and advances in computing and communication technologies.
The people's future living environments are emerging based upon information resource provided by the connections of various communication networks for users. New small devices like Personal Digital Assistants (PDAs), mobile phones, handhelds, and wearable computers enhance information processing and ubiquitous access of users.
Mobile networking is one of the most important technologies supporting pervasive computing. During the last decade, advances in both hardware and software techniques have resulted in mobile hosts and wireless networking.
Generally there are two distinct approaches for enabling wireless mobile units to communicate with each other: *Infrastructured* and *Infrastructureless*.
Wireless mobile networks have traditionally been based on the cellular concept and relied on good infrastructure support, in which mobile devices communicate with access points (or base stations) connected to the fixed network infrastructure. Typical examples of this kind of wireless networks are GSM, UMTS, WLL, WLAN, etc. [13, 38, 52, 82]
In recent years the widespread availability of wireless communication and handheld devices has stimulated research on self-organizing networks that do not require a pre-established infrastructure. These *ad hoc networks*, as they are commonly called, consist of autonomous nodes that collaborate in order to transport information. Usually, these nodes act as end systems and routers at the same time.
Ad hoc networks can be subdivided into two classes: *static* and *mobile*. In static ad hoc networks the position of a node may not change once it has become part of the network. Typical examples are rooftop networks [85].
In mobile ad hoc networks, systems may move arbitrarily.
A Mobile Ad Hoc Network is commonly called a MANET. A MANET is a collection of wireless mobile nodes that dynamically form a network to exchange information without using any pre-existing fixed network infrastructure or a centralized administration.
MANET nodes are equipped with wireless transmitters and receivers using antennas, which may be omni-directional (broadcast), highly-directional (point-to-point), possibly steerable, or some combination thereof. At a given point in time, depending on the nodes' positions and their transmitter and receiver coverage patterns, transmission

power levels and co-channel interference levels, a wireless connectivity in the form of a random, multihop graph or “ad hoc” network exists between the nodes. This ad hoc topology may change with time as the nodes move or adjust their transmission and reception parameters.
In such an environment, it may be necessary for one mobile host to enlist the aid of other hosts in forwarding a packet to its destination, due to the limited range of each mobile host’s wireless transmissions.
The following assertion is extracted from RFC 2501 [5]: "The vision of mobile ad hoc networking is to support robust and efficient operation in mobile wireless networks by incorporating routing functionality into mobile nodes. Such networks are envisioned to have dynamic, sometimes rapidly-changing, random, multihop topologies".
MANETs are a very important part of communication technology that supports truly pervasive computing, because in many contexts information exchange between mobile units cannot rely on any fixed network infrastructure, but on rapid configuration of wireless connections on-the-fly.
Next generation of mobile communications will include both prestigious infrastructured wireless networks and infrastructureless Mobile Ad Hoc Networks (MANETs).
Examples where mobile ad hoc networks may be employed are the establishment of connectivity among handheld devices or between vehicles.
The special features of MANET bring this technology great opportunities together with severe challenges. Since mobile ad hoc networks change their topology frequently and without prior notice, routing in such networks is a challenging task.

A MANET is some way like an autonomous system in which mobile hosts connected by wireless links are free to move randomly and often act as routers at the same time. The traffic types in ad hoc networks are quite different from those in an infrastructured wireless network, including:
1) Peer-to-Peer. Communication between two nodes that are within one hop. Network traffic (Bps) is usually consistent.
2) Remote-to-Remote. Communication between two nodes beyond a single hop but which maintain a stable route between them. This may be the result of several nodes staying within communication range of each other in a single area or possibly moving as a group. The traffic is similar to standard network traffic.
3) Dynamic Traffic. This occurs when nodes are dynamic and moving around. Routes must be reconstructed. This results in a poor connectivity and network activity in short bursts.

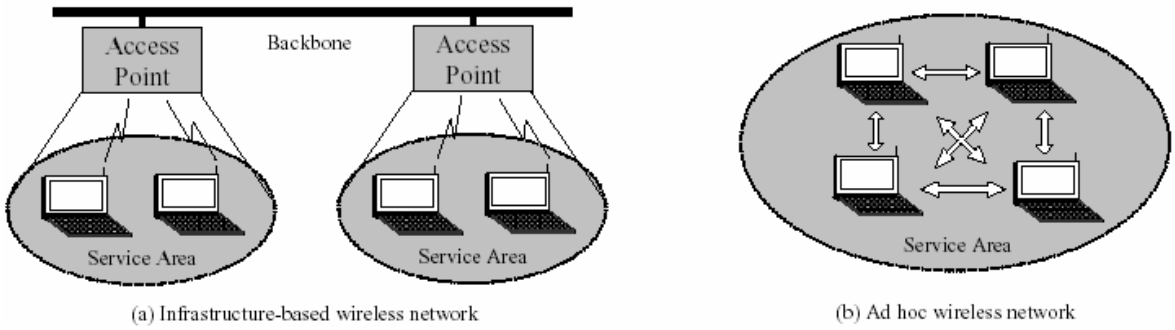


Figure 1.1. *Examples of both infrastructured and infrastructureless ad hoc wireless networks.*

A MANET has the following features:

1) Autonomous terminal.
In a MANET, each mobile terminal is an autonomous node, which may function as both a host and a router. In other words, besides the basic processing ability as a host, the mobile nodes can also perform switching functions as a router. So usually endpoints and switches are indistinguishable in MANET.

2) Distributed operation.
Since there is no background network for the central control of the network operations, the control and management of the network is distributed among the terminals. The nodes involved in a MANET should collaborate amongst themselves and each node acts as a relay as needed, to implement functions e.g. security and routing.

3) Multihop routing.
Basic types of ad hoc routing algorithms can be single-hop and multihop, based on different link layer attributes and routing protocols. Single-hop MANET is simpler than multihop in terms of structure and implementation, with the cost of lesser functionality and applicability. When delivering data packets from a source to its destination out of the direct wireless transmission range, the packets should be forwarded via one or more intermediate nodes.

4) Dynamic network topology.
Since the nodes are mobile, the network topology may change rapidly and unpredictably and the connectivity among the terminals may vary with time. MANET should adapt to the traffic and propagation conditions as well as the mobility patterns of the mobile network nodes. The mobile nodes in the network dynamically establish routing among

themselves as they move about, forming their own network on the fly. Moreover, a user in the MANET may not only operate within the ad hoc network, but may require access to a public fixed network.

5) Fluctuating link capacity.
The nature of high bit-error rates of wireless connection might be more profound in a MANET. One end-to-end path can be shared by several sessions. The channel over which the terminals communicate is subject to noise, fading, and interference, and has less bandwidth than a wired network. In some scenarios, the path between any pair of users can traverse multiple wireless links and the link themselves can be heterogeneous.

6) Light-weight terminals.
In most cases, the MANET nodes are mobile devices with less CPU processing capability, small memory size, and low power storage. Such devices need optimized algorithms and mechanisms that implement the computing and communicating functions.

Subsuming, a mobile ad hoc network is a collection of autonomous mobile nodes that form a dynamic, purpose-specific, multi-hop radio network in a decentralized fashion. The peculiarity of such networks is the conspicuous absence of a fixed support infrastructure such as mobile switching centres, base stations, access points, and other centralized machinery traditionally seen in wireless networks. The network topology is constantly changing as a result of nodes joining in and moving out. Packet forwarding, routing, and other network operations are carried out by the individual nodes themselves.
The features of MANET introduce several challenges that must be studied carefully before a wide commercial deployment can be expected. These include:

1) Routing
Since the topology of the network is constantly changing, the issue of routing packets between any pair of nodes becomes a challenging task.
In a MANET, routers (i.e. hosts) can be mobile, and inter-router connectivity can change frequently during normal operation.
In contrast, the Internet, like nearly all telecom networks, possesses a quasi-fixed infrastructure consisting of routers or switches that forward data over hardwired links. Traditionally, end-user devices, such as host computers or telephones, attach to these networks at fixed locations. As a consequence, they are assigned addresses based on their location in a fixed network-addressing hierarchy and oftentimes assume an identity equivalent to their address. This identity-location equivalence greatly simplifies routing in these systems, as a user's location does not change.
Increasingly, end devices are mobile, meaning that they can change their point of

attachment to the fixed infrastructure. This is the paradigm of cellular telephony and its Internet equivalent, mobile IP. In this approach, a user's identity depends upon whether the user adopts a location-dependent (temporary) or location-independent (permanent) identifier.

Users with temporary identifiers are sometimes referred to as nomadic, whereas users with permanent identifiers are referred to as mobile.

The distinction is that although nomadic users may move, they carry out most network-related functions in a fixed location. Mobile users, must work "on the go," changing points of attachment as necessary. In either case, additional networking support may be required to track a user's location in the network so that information can be forwarded to its current location using the routing support within the more traditional fixed hierarchy.

Supporting this form of host mobility (or nomadicity) requires address management, protocol interoperability enhancements and the like, but core network functions such as hop-by-hop routing still presently rely upon pre-existing routing protocols operating within the fixed network.

Internet is hardly tuned to allow mobility in the midst of data transfers because protocols are not conceived for devices that frequently change their point of attachment in the topology. There is typically a change of the physical IP address each time a mobile node changes its point of attachment and thus its reachability to the Internet topology. This results in losing packets in transit and breaking transport protocols connections if mobility is not handled by specific services. The protocol stack must therefore be upgraded with the ability to cross networks in the midst of data transfers, without breaking the communication session and with minimum transmission delays and signalling overhead. This is commonly referred to as *mobility support*.

Host mobility support is handled by Mobile IPv6.

In contrast, the goal of mobile ad hoc networking is to extend mobility into the realm of autonomous, mobile, wireless domains, where a set of nodes, which may be combined routers and hosts, themselves form the network routing infrastructure in an ad hoc fashion.

With Mobile Ad Hoc Networking, the routing infrastructure can move along with the end devices. Thus the infrastructure's routing topology can change, and the addressing within the topology can change. In this paradigm, an end-user's association with a mobile router (its point of attachment) determines its location in the MANET. As before, a user's identity may be temporary or permanent. But now, given the fundamental change in the composition of the routing infrastructure (that is, from fixed, hardwired, and bandwidth-rich to dynamic, wireless, and bandwidth-constrained), much of the fixed infrastructure's control technology is no longer useful.

The infrastructure's routing algorithms and, indeed, much of the networking suite must be reworked to function efficiently and effectively in this mobile environment.

Multicast routing is another challenge because the multicast tree is no longer static due to the random movement of nodes within the network. Routes between nodes may potentially contain multiple hops, which is more complex than the single hop communication.

2) Security

In addition to the common vulnerabilities of wireless connection, an ad hoc network has its particular security problems [52, 62, 79, 80, 86, 87].

Mobile hosts "join" in, on the fly, and create a network on their own. With the network topology changing dynamically and the lack of a centralized network management functionality, these networks tend to be vulnerable to a number of attacks. If such networks are to succeed in the commercial world, the security aspect naturally assumes paramount importance.

The unreliability of wireless links between nodes, constantly changing topology owing to the movement of nodes in and out of the network and lack of incorporation of security features in statically configured wireless routing protocols not meant for ad hoc environments all lead to increased vulnerability and exposure to attacks.

Security in wireless ad hoc networks is particularly difficult to achieve, notably because of the vulnerability of the links, the limited physical protection of each of the nodes, the sporadic nature of connectivity, the dynamically changing topology, the absence of a certification authority and the lack of a centralized monitoring or management point. This, in effect, underscores the need for intrusion detection, prevention, and related countermeasures.

The wireless links between nodes are highly susceptible to link attacks, which include passive eavesdropping, active interfering, leakage of secret information, data tampering, impersonation, message replay, message distortion, and denial of service. Eavesdropping might give an adversary access to secret information, violating confidentiality. Active attacks might allow the adversary to delete messages, to inject erroneous messages, to modify messages, and to impersonate a node, thus violating availability, integrity, authentication, and non-repudiation (these and other security needs are discussed in the next section).

Ad hoc networks do not have a centralized piece of machinery such as a name server, which could lead to a single point of failure and thus, make the network that much more vulnerable. In contrast, however, the lack of support infrastructure and the possibilities of new types of prior context may prevent the application of standard techniques for key agreement. This gives rise to a need for schemes to ensure key agreement.

An additional problem related to the compromised nodes is the potential *byzantine failures* encountered within Mobile Ad hoc Network (MANET) routing protocols wherein a set of the nodes could be compromised in such a way that the incorrect and malicious behaviour cannot be directly noted at all.

The compromised nodes may seemingly operate correctly, but, at the same time, they may make use of the flaws and inconsistencies in the routing protocol to undetectably distort the routing fabric of the network. In addition, such malicious nodes can also create new routing messages and advertise nonexistent links, provide incorrect link state information, and flood other nodes with routing traffic, thus inflicting byzantine failures on the system. Such failures are severe, especially because they may come from seemingly trusted nodes, the malicious intentions of which have not yet been noted. Even if the compromised nodes were noticed and prevented from performing incorrect actions, the erroneous information generated by the byzantine failures might have already been propagated through the network.
No part of the network is dedicated to support individually any specific network functionality, with routing (topology discovery, data forwarding) being the most prominent example. Additional examples of functions that cannot rely on a central service, and that are also of high relevance, are naming services, certification authorities, directory, and other administrative services. Even if such services were assumed, their availability would not be guaranteed, either due to the dynamically changing topology that could easily result in a partitioned network, or due to congested links close to the node acting as a server.

Every node that comes into the transmission range of an Ad Hoc network physically becomes a part of the network. Logically, however, the Ad Hoc network may be only partially open or even closed to visitors. Furthermore, nodes that are a part of a network can have different classification levels; accordingly, an Ad Hoc network could have routes with different levels of sensitivity and security. A multilevel communication model could address this issue.
Lastly, scalability is another issue that has to be addressed when security solutions are being devised, for the simple reason that an ad hoc network may consist of hundreds or even thousands of nodes. Many ad hoc networking protocols are applied in conditions where the topology must scale up and down efficiently, e.g., due to network partitions or mergers. The scalability requirements also directly affect the scalability requirements targeted to various security services such as key management.

Standard security solutions would not be good enough since they are essentially for statically configured systems. This gives rise to the need for security solutions that adapt to the dynamically changing topology and movement of nodes in and out of the network.

3) Reliability
Wireless link characteristics introduce also reliability problems, because of the limited wireless transmission range, the broadcast nature of the wireless medium (e.g. hidden

terminal problem, fading), mobility-induced packet losses, and data transmission errors.

4) Quality of Service (QoS)

The goal of QoS support is to achieve a more deterministic communication behaviour, so that information carried by the network can be better preserved and network resources can be better utilized. Intrinsic to the notion of QoS is an agreement or a guarantee by the network to provide a set of measurable pre-specified service attributes to the user in terms of network delay, delay variance (jitter), available bandwidth, probability of packet loss (loss rate), and so on. The QoS forum defines QoS as *the ability of a network element (e.g., an application, a host, or a router) to provide some level of assurance for consistent network data delivery.* The IETF RFC 2386 characterizes QoS as a set of service requirements to be met by the network while transporting a packet stream from source to destination.

A network's ability to provide a specific QoS depends upon the inherent properties of the network itself, which span over all the elements in the network. For the transmission link, the properties include link delay, throughput, loss rate, and error rate. For the nodes, the properties include hardware capability, such as processing speed and memory size. Above the physical qualities of nodes and transmission links, the QoS control algorithms operating at different layers of the network also contribute to the QoS support in networks. Unfortunately, the inherent features of MANETs show weak support for QoS. The wireless link has low, time-varying raw transmission capacity with relatively high error rate and loss rate. In addition, the possible various wireless physical technologies that nodes may use simultaneously to communicate make MANETs heterogeneous in nature. Each technology will require a MAC layer protocol to support QoS. Therefore, the QoS mechanisms above the MAC layer should be flexible to fit the heterogeneous underlying wireless technologies.

Providing different quality of service levels in a constantly changing environment will be a challenge. The inherent stochastic feature of communications quality in a MANET makes it difficult to offer fixed guarantees on the services offered to a device. An adaptive QoS must be implemented over the traditional resource reservation to support the multimedia services.

5) Internetworking

In addition to the communication within an ad hoc network, internetworking between MANET and fixed networks (mainly IP based) is often expected in many cases. The coexistence of routing protocols in such a mobile device is a challenge for the harmonious mobility management.

6) Power Consumption

For most of the light-weight mobile terminals, the communication-related functions

should be optimised for lean power consumption. Conservation of power and power-aware routing must be taken into account.

1.1 Classification of Ad Hoc Routing Protocols

We can distinguish two main different approaches for routing in MANETs environments: *topology-based* and *position-based* routing.
Topology-based routing protocols use the information about the links that exist in the network to perform packet forwarding. They can be further divided into *proactive*, *reactive* and *hybrid* approaches.
Position-based (or geographic) routing requires that information about the physical position of the participating nodes be available, it then avoids overhead by requiring only accurate neighbourhood information and a rough idea of the position of the destination.
Topology-based routing approaches are discussed below while position-based routing is the subject of chapter 4.

Topology-based routing protocols use the information about the links that exist in the network to perform packet forwarding.
They can be further divided into *proactive (table-driven), reactive (on-demand)* and *hybrid* approaches.

Proactive algorithms may employ classical routing strategies such as distance-vector routing (e.g., DSDV [3, 4]) or link-state routing (e.g., OLSR and TBRPF [3, 4]). They maintain routing information about the available paths in the network even if these paths are not currently used. The main drawback of these approaches is that the maintenance of unused paths may occupy a significant part of the available bandwidth if the topology of the network changes frequently.

In response to this observation, reactive routing protocols were developed (e.g., DSR [6], TORA, and AODV [3, 4]).
Reactive routing protocols maintain only the routes that are currently in use, thereby reducing the burden on the network when only a small subset of all available routes is in use at any time. However, they still have some inherent limitations. First, since routes are only maintained while in use, it is typically required to perform a route discovery

before packets can be exchanged between communication peers. This leads to a delay for the first packet to be transmitted. Second, even though route maintenance for reactive algorithms is restricted to the routes currently in use, it may still generate a significant amount of network traffic when the topology of the network changes frequently. Finally, packets en route to the destination are likely to be lost if the route to the destination changes.

Hybrid ad hoc routing protocols such as ZRP [3, 4] combine local proactive routing and global reactive routing in order to achieve a higher level of efficiency and scalability. However, even a combination of both strategies still needs to maintain at least those network paths that are currently in use, limiting the amount of topological changes that can be tolerated within a given amount of time.

An interesting and more accurate taxonomy of topology-based Ad Hoc routing protocols has been introduced by Feeney [4].
This taxonomy is based on the division of the protocols according to following criteria, reflecting fundamental design and implementation choices:

- *Communication model*. What is the wireless communication model? Multi- or single-channel?
Protocols can be divided according to communications model to protocols that are designed for *multi-channel* or *single-channel* communications. Multi-channel protocols are routing protocols generally used in TDMA or CDMA-based networks. They combine channel assignment and routing functionality.
Single-channel protocols presume one shared media to be used. They are generally CSMA/CA-oriented, but they have a wide diversity in which extent they rely on specific link-layer behaviours.

- *Structure*. Are all nodes treated uniformly? How are distinguished nodes selected? Is the addressing hierarchical or flat?
Structure of a network can be classified according to node uniformity. Some protocols treat all the nodes uniformly, other make distinctions between different nodes. In *uniform protocols* there is no hierarchy in network, all nodes send and respond to routing control messages at the same manner.
In *non-uniform protocols* there is an effort to reduce the control traffic burden by separating nodes in dealing with routing information. Non-uniform protocols fall into two categories: protocols in which each node focuses routing activity on a subset of its neighbours and protocols in which the network is topologically partitioned. These two different methods for non-uniformity are called *neighbour selection* and *partitioning* respectively.

With neighbour selection mechanism, every node has its own criteria to classify network nodes to near or to remote nodes. In partitioning protocols that differentiation is to use hierarchical node separation.
Hierarchical protocols have some upper-level and lower-level nodes and certain information difference between them.

- *State Information*. Is network-scale topology information obtained at each node?
Protocols may be described in terms of the state information obtained at each node and/or exchanged among nodes. Some use the principle that every node in a network maintains large-scale topology information. This principle is just the same as link-state protocols use.
Other protocols do not maintain large-scale topology information. They only may maintain topology information needed to know the nearest neighbours. The best known of such protocols are distance-vector protocols, which maintain a distance and a vector to a destination (hop count or other metric and next hop).

- *Scheduling*. Is route information continually maintained for each destination?
The way to obtain route information can be a continuous or a regular procedure or it can be trigged only by on demand. On that basis the protocols can be classified to proactive and on-demand protocols. *Proactive protocols*, which are also know as table-driven protocols, maintain all the time routing information for all known destinations at every source. In these protocols nodes exchange route information periodically and / or in response to topology change.
In on-demand i.e. in *reactive protocols* the route is only calculated on demand basis. That means that there is no unnecessary routing information maintained. The route calculation process is divided to a route discovery and a route maintenance phase. The route discovery process is initiated when a source needs a route to a destination.
The route maintenance process deletes failed routes and re-initiates route discovery in the case of topology change.

This model does not take into account if a protocol is unicast, multicast, geocast or broadcast. Also the taxonomy doesn't deal with the question how the link or node related costs are measured. These properties are however worth to be considered in classification and evaluating applicability of protocols.
Based on that lack the taxonomy has been slightly modified [3] by adding such features as *type of cast* and *cost function*. Type of cast feature is an upper level classification and so the protocols to be classified must be firstly divided by type of cast and after that the more accurate taxonomy can be applied. The aforementioned taxonomy [4] is applied to unicast protocols, while in the context of multicast and geocast protocols a specified taxonomy has been introduced [3].

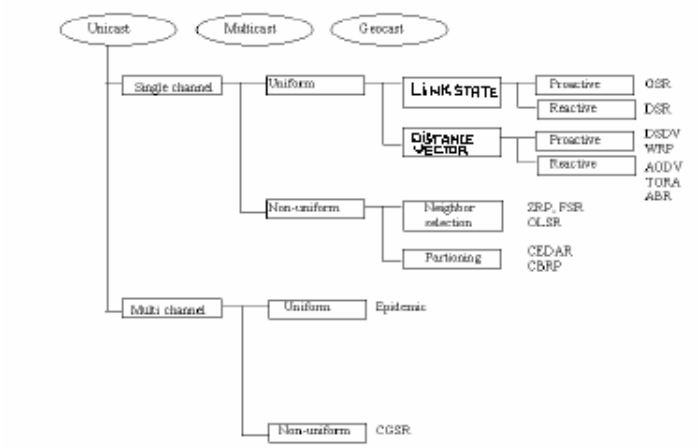


Figure 1.2*. Taxonomy of routing protocols. Classification of unicast protocols.*

Different kind of ad hoc routing protocols are suitable for different kind of network structures and nodes' behaviours. When evaluating protocols one needs some appropriate classification also for the features of performance metrics.
The critical features for ad hoc networks can be classified to following quantitative and qualitative features. These features determine the networking context in which a protocol must work and its performance is measured.
Quantitative features are:
- *Network size.* Measured in number of nodes.
- *Network connectivity*. The average degree (number of neighbours) of a node.
- *Topological rate of change*. The speed with which a network's topology is changing.
- *Link capacity*. Effective link speed measured in bits/second, after accounting for losses due to multiple access, coding, framing, etc.
- *Fraction of unidirectional links.* This is due to the characteristic of radio waves and receivers.
- *Fraction and frequency of sleeping nodes*.
- *Network settling time*, which is the time for a network to reach a stable state and be able to send its first message reliably.
- *Network join time*, which is the time for an entering node or group of nodes to become integrated into the ad hoc network.
- *Network depart time*, which is the time required for the ad hoc network to recognize the loss of one or more nodes, and reorganize itself to manage lacking links.
- *Network recovery time*, which is the time for a network to recover after a condition that dictates reorganization of the network.

- *Frequency of updates*, which is the number of control packets or overhead bytes inside packets to be sent in a given time to maintain proper network operation. This means also same as overhead.
- *Memory required*, which is the storage space required for routing tables and other management tables.
- *Network scalability number*, which is the number of nodes that a network can scale to and still preserve communications.

Network recovery time is an important factor for fast changing dynamic networks. If the recovery time is too long, it causes the network to maintain for a too long time an unstable state. That causes routing errors to happen, which cause lost packets and needs for retransmissions.
Frequency of updates is also a meaningful parameter for bandwidth constrained radio networks. If the protocol needs too often or too large update packets to be sent, it will consume in dynamic networks too much available total capacity.
Network scalability number has a meaning when there is a need for large scale networks to be constructed. The large scale is not a clear term, but the number of nodes can surprisingly grow up, when ad hoc environments reach their success. In military environments scalability is an essence.

According to RFC 2501 [5] quantitative metrics for network *routing protocol* performance are:
- End-to-end data *throughput* and *delay*. These are measures of a routing policy's effectiveness - how well it does its job- as measured from the 'external' perspective of other policies that make use of routing.
- *Route acquisition time*. The time required to establish route(s) when requested. It is a particular concern for on-demand protocols. Routing updates should be delivered in a timely fashion. Update messages that arrive late may not reflect the true state of links or routers on the network. They can cause incorrect forwarding or even propagate false information and weaken the credibility of the update information. If a node that relays information between two large connected components is advertised as "down" by malicious neighbours, large parts of the network become unreachable.
- *Percentage out-of-order delivery*. An external measure of connectionless routing's performance. It can affect how efficiently transport layer protocols can perform their task.
- *Efficiency*. It is an internal measure of protocols effectiveness. It deals with the protocol overhead questions. It could be said to be some kind of utilization ratio between routing effectiveness and overhead.
Some ratios that measure the 'internal' efficiency of a protocol are:

- Average number of data bits transmitted / data bits delivered

- Average number of control bits transmitted / data bit delivered. This measures the bit efficiency of the protocol in expending control overhead to delivery data.
- Average number of control and data packets transmitted / data packet delivered. This measure tries to capture a protocol's channel access efficiency, as the cost of channel access is high in contention-based link layers.

The qualitative critical features are the following:
- *Loop freedom*. Generally desirable to avoid problems such as packet spinning around in the network for arbitrary time periods. Solutions such as TTL values can bound the problem, but a more structured and well-formed approach is generally desirable as it usually leads to better overall performance.
- *Demand based operation*.
- *Proactive operation*. Generally performs worse than on demand approach due to too many control packets, but in certain contexts, the additional latency demand-based operation incurs may be unacceptable. If bandwidth and energy resources permit, proactive operation is desirable in these contexts.
- *Security*. Without some form of network-level or link-layer security, a MANET routing protocol is vulnerable to many forms of attack. Some desirable properties a routing protocol should have are:
  -- *Confidentiality*
  -- *Integrity*.
  -- *Availability*.
  -- *Self-Stabilization*. A routing protocol should be able to automatically recover from any problem in a finite amount of time without human intervention. That is, it must not be possible to permanently disable a network by injecting a small number of malformed packets. If the routing protocol is self-stabilizing, an attacker who wishes to inflict continuous damage must remain in the network and continue sending malicious data to the nodes, which makes the attacker easier to locate.
  -- *Byzantine Robustness*. A routing protocol should be able to function correctly even if some of the nodes participating in routing are intentionally disrupting its operation. Byzantine robustness can be seen as a stricter version of the self-stabilization property; the routing protocol must not only automatically recover from an attack, it should not cease functioning even during the attack. Clearly, if a routing protocol does not have the self stabilization property, it cannot have byzantine robustness either.
  -- *Isolation*. Isolation requires that the protocol be able to identify misbehaving nodes and render them unable to interfere with routing. Alternatively, the routing protocol should be designed to be immune to malicious nodes.

- *Knowledge of nodal locations*. Does the routing algorithm require local or global knowledge of the network?
- *Effect to topology changes*. Does the routing algorithm need complete restructuring or incremental updates?
- *Adaptation to radio communications environment.* Do nodes use estimation knowledge of fading, shadowing or multi-user interference on links in their routing decisions?
- *Sleep period operation*. As a result of energy conservation, or some other need to be inactive, nodes of a MANET may stop transmitting and/or receiving (even receiving requires power) for arbitrary time periods. A routing protocol should be able to accommodate such sleep periods without overly adverse consequences. This property may require close coupling with the link-layer protocol through a standardized interface.
- *Power Consciousness*. Does the network employ routing mechanisms that consider the remaining battery life of a node?
- *Single- or multi- channel.* Does the routing algorithm utilize a separate control channel?
- *Bidirectional or unidirectional links support*. Does the routing algorithm perform efficiently on unidirectional links? Unidirectional links in ad hoc environment are not exceptions, because of asymmetrical nature of radio channel caused by interference, jamming and different receiver or transmitter characteristics.
- *QoS routing and handling of priority messages*. Does the routing algorithm support priority messaging and reduction of latency for delay sensitive real time traffic? Can the network send priority messages even when it is overloaded with routine traffic levels?
- *Real-time voice and video services*. Can the network support simultaneously real-time multicast voice and/ or video on-demand services while supporting other routine traffic services? Applications to need these services will emerge most probably in all ad hoc network solutions, so the implemented routing method should support that need.
- *Ability to use multiple routes*. This is in order to avoid congestion.

Chapter 2

Vehicular Ad Hoc Networks: properties and challenges

A Mobile Ad Hoc Network for Inter-Vehicle Communications (IVC) is also called a Vehicular Ad Hoc Network (VANET) [74]. It is a network whose nodes are vehicles equipped with hosts and wireless transmitters.

Communication between vehicles by means of wireless technology has a large potential to improve traffic safety and travel comfort of drivers and passengers. Current advances in the field of wireless ad hoc networks show that inter-vehicle communication based on vehicular ad hoc networks is a feasible approach that has a competitive edge over cellular network-based telematics with respect to several aspects:

- low data transport times for emergency warnings,
- low costs for usage due to the absence of a central administration and the use of unlicensed frequency bands,
- robustness due to the network's mesh structure for several scenarios.

Communication between vehicles is considered a prime area where mobile ad hoc networks are likely to be deployed in the near future. The reasons for this focus on VANETS are multiple:

- First, there is a wealth of *applications* for ad-hoc communication between vehicles, ranging from emergency warnings and distribution of traffic as well as road condition information to chatting and distributed games [46, 63, 67, 68, 75, 84, 87].

- Second, many vehicle manufacturers and their suppliers are actively supporting research on how to integrate mobile ad-hoc networks into vehicles. They believe VANETs will bring great benefits and advantages to future transportation. Thus, vehicle manufacturers invest their money in research, while other projects for ad hoc networks do not have this financial power behind them [75, 54, 83, 84].

- Vehicles and roads' security involve governments and public entities with great interest, thus they also promote the research in vehicular networks. There is a trend from passive toward *intelligent, active* safety systems. Passive safety systems include: conventional ABS, passive airbag, ABS with traction control, etc.
  Active safety system include: adaptive airbag, adaptive cruise control, electronic stability program, etc...
  But wireless communications among vehicles will enable *active intelligent* safety mechanism: intelligent on-board systems for active safety application processing, coupled with wireless communications for real time access to relevant off-board data will enhance planned active safety applications and will enable new safety applications [75, 46, 54, 59, 63, 36, 68, 74, 83, 84].

Furthermore, lots of aspects that make problematic the development of Ad Hoc networks in other environments are not a constraint in vehicular networks:

- first, vehicles can easily provide the required power for wireless communication, and adding some weight for antennas and additional communication hardware does not cause major problems. Some technologies like Dedicated Short-Range Communication (DSRC, it will be standardized as 802.11p) [64, 63, 75] or 3G cellular technologies [13, 82] can be utilized to transmit packets among vehicles. DSRC is a solution created ad hoc for inter-vehicles communications; modified UMTS technologies could also be utilized: UMTS Terrestrial Radio Access with Time Division Duplexing (UTRA-TDD) [13, 82, 76] which supports communication range of more than 1 km, is adopted by FleetNet project as the radio interface.

- Second, it can be expected that vehicles will have an accurate knowledge of their own geographic position, e.g., by means of GPS [12, 14, 58].
  They also will be equipped with accurate digital road maps and then they will know roads' topology with great accuracy.
  Therefore many aspects making problematic the deployment of ad-hoc networks in other scenarios are not relevant here.

- Third, vehicles will be equipped with high power hosts, so computation power and storage space are non constraints to inter-vehicle communications development. Thus, algorithms can be focused on efficiency rather than on power and computation awareness.

As a consequence, many vehicle manufacturers are sustaining an active research that aim to the development and deployment of ad hoc networks among vehicles.
There is a strong desire that brings toward the implementation of wireless inter-vehicles communication, but there are many problems to face and many proposals and solutions are open fields of research, where the followings are indeed relevant aspects:

- the design of communication protocols in the framework of inter-vehicle communication (IVC) is extremely challenging due to the variety of application requirements and the *tight coupling between an application and its supporting protocols* [46, 51].

- Highly mobile nodes make up the network. Network's topology is dynamic and rapidly changing.
  Especially for highway scenarios, network's topology can be not dense and routing must adapt to such topology. On the other side, in city environments vehicles do not have high velocities and network's topology can be very dense. Buildings can introduce several transmission problems, thus routing algorithms must work in heterogeneous environments. Routing can benefits from detailed digital road maps.

- Protocols can be optimized by taking into account the special features of vehicles.
  For example, GPS, Geographic Information System (GIS), and digital map can help a node to be aware of its location and the surrounding, like road topology. Since the road topology somewhat implies the network topology in IVC, this knowledge does help to make the routing protocol more efficient.
  Vehicles have a predictable, high mobility that can be exploited for system optimization.
  The motion of vehicles can significantly affect message delivery latency.

- Considering the application requirements for IVC, broadcast routing that disseminates information to a set of nodes that could be far from each other seems to be a necessary supporting mechanism; it could be optimized according to the requirement of an application.
  On the contrary, unicast routing might be superfluous in most cases.
  This is especially true for security applications.
  Alarms emitted from security applications are relevant only in a defined geographic area, thus those alarms should remain in a defined area and for a given interval of time to alert all vehicles approaching the dangerous area. What is very important for security applications is a reliable *Geocast* (chapter 5) that

delivers a message to all the vehicles in a given geographic area. This mechanism should be enhanced to support a time-stable geocast, a geocast that after delivering a message to a given area maintains that message in the area for a specified time interval, so that it is received by all mobile nodes that enter the area but where not present when the message arrived in the area for the first time.

- Position will be first class information: a lot of applications will be based on the positions of the hosts and routing protocols will utilize the positions to deliver packets. For example, safety applications will deliver alert signals to all nodes in a given area, regardless of nodes identities and traffic control applications will require data from vehicles located in specific areas [53, 54, 74, 75, 83, 84].

- Benefiting from the large capacities (in terms of both space and power) of vehicles, the nodes of these networks can have long transmission ranges and virtually unlimited lifetimes.

Inter-vehicle communications open the door for a plethora of applications and services ranging from automated highway systems to distributed passengers teleconference. These applications may be classified to safety and non-safety applications. Under safety applications, Vehicle Collision Avoidance (VCA) has attracted considerable attention since it is directly related to minimizing number of accidents on the road[75, 53, 54, 63, 67, 74, 83].
Some potential Vehicle-to-Vehicle safety applications scenarios are the followings [75]:

- Cooperative Collision Warning
- Highway Merge Assistant
- Highway/Rail Collision Warning
- Approaching Emergency Vehicle Warning
- Blind Merge Warning
- Vehicle-to-Vehicle Road Feature Notification
- Visibility Enhancer
- Cooperative Adaptive Cruise Control
- Lane Change Assistant
- Pre-Crash Sensing
- Post-Crash Warning
- SOS Services
- Instant Messaging

For example, LVMM, the middleware described in this document, could be utilized to support Cooperative Adaptive Cruise Control and Instant messaging applications because it allows low-latency multicast communications between vehicles with similar mobilities.

On the other hand, non-safety applications may include real-time road traffic estimation for trip planning, high-speed tolling, collaborative expedition, information retrieval, and entertainment applications.

While multi-hop ad hoc network communication among vehicles provides for many interesting and important applications (e.g., travelling safety, smoothed traffic flow), users also will be interested in accessing Internet services. This access can be achieved via roadside installed Internet gateways [78, 76, 54].

There is a desire to connect permanently to the Internet from anywhere, at anytime, without any disruption of service, particularly for those who spend a significant amount of their time in transportation systems. Connecting vehicles to the Internet is a means to achieve this trend toward ubiquitous computing. So far, conventional technologies have been providing information to drivers and passengers via signboards, radio systems, mobile phones and more sophisticated technologies. With the advent of the Internet and IPv6 [9, 10], these systems can be integrated into a general digital communication system. Once connected to the Internet, not only Internet data can be accessed from the vehicles, but vehicles can also be monitored and various environmental information generated by the embedded sensors can be advertised to the Internet.

The Internet integration of inter-vehicle communication systems entails several difficulties, such as mobility support, communication efficiency, or the discovery and handover of connections from one gateway to the next.

In addition, it is envisioned that VANETs may play an important role in improving the capacity and coverage of future wireless networks via:

i) complementing the existing cellular infrastructure in hot spot areas where the system gets overloaded and it may be favourable for vehicles to assist one another in reaching the base station (BS) (via multi-hopping) rather than continuously competing to access the uplink and

ii) extending the coverage of the cellular infrastructure via enabling an out-of-range vehicle to forward its data through multiple hops until a BS is reachable.

As a final summary, an extract from the VANET [74] homepage is quoted below.

"Creating high-performance, highly scalable, and secure VANET technologies presents an extraordinary challenge to the wireless research community. Yet, certain limitations commonly assumed in ad hoc networks are mitigated in VANET. For example, VANET may marshal relatively large computational resources. Ample and recharging power

sources can be assumed. Mobility patterns are constrained by road paths and driving speed restrictions. VANET represents high resource/performance wireless technology. As such, VANET can use significantly different approaches than sensor networks.

VANET applications will include on-board active safety systems leveraging vehicle-vehicle or roadside-vehicle networking. These systems may assist drivers in avoiding collisions. Non-safety applications include real-time traffic congestion and routing information, high-speed tolling, mobile infotainment, and many others. "

Furthermore, the challenge is to come up with software capable of operating in real time. In other words, it must transmit data without delay so that the information reaches all those involved in good time.

2.1 DSRC - Dedicated Short Range Communications

Lots of in progress researches are studying the problem of inter-vehicles communications at the physical- and link-layers.
Several MAC- and physical-level technologies exist for wireless communications, which guarantee high reachability: technology like 802.11b, with 2.4GHz in ISM band, when utilized for inter-vehicles communications allows reaching nodes up to 600 metres and makes the connection between two oncoming cars driving at a speed of 130km/h in opposite directions lasts about 5s. [49]
However, it is worth considering the fact that instead of 802.11b with 2.4GHz, technologies like DSRC, with 5.4GHz in the U.S.A., will be utilized [75].
5.9 GHz DSRC (Dedicated Short Range Communications) is a short to medium range communications service that supports both Public Safety and Private operations in roadside to vehicle and vehicle to vehicle communication environments. DSRC is meant to be a complement to cellular communications by providing very high data transfer rates in circumstances where minimizing latency in the communication link and isolating relatively small communication zones are important. [88]
DSRC is 75 MHz of spectrum at 5.9 GHz allocated by the Federal Communications Commission (FCC) in the U.S.A. to "increase traveller safety, reduce fuel consumption and pollution, and continue to advance the nation's economy." This promising development is designed to support low-latency vehicle to vehicle and vehicle-to-infrastructure wireless communication using a variant of the IEEE 802.11a technology. DSRC will support safety critical communications, such as collision warnings, as well as other valuable Intelligent Transportation System applications, such as Electronic Toll Collection (ETC), real-time traffic advisories, digital map update, etc. The versatility of DSRC greatly enhances the likelihood of its deployment by various industries and adoption by consumers.
DSRC is under standardization by the IEEE into the 802.11p technology.
The physical layer (PHY) of DSRC is adapted from IEEE 802.11a PHY based on orthogonal frequency division multiplex (OFDM) technology. Moreover, the multiple access control (MAC) layer of DSRC is very similar to the IEEE 802.11 MAC based on the carrier sense multiple access with collision avoidance (CSMA/CA) protocol with some minor modifications.
The 5.9 GHz band consists of seven ten-megahertz channels which includes one control channel and six service channels. DSRC, which involves vehicle-to-vehicle and vehicle-to-infrastructure communications, is expected to support both safety and non-safety applications. However, priority is given to safety applications since the non-public safety use of the 5.9 GHz band would be inappropriate if it leads to degrading the performance of safety/public safety applications [FCC]. This is attributed to the fact that

safety applications are meant to save lives via warning drivers of an impending dangerous condition or event in a timely manner in order to take corrective actions. Therefore, response time and reliability are basic requirements of safety applications as discussed later in the paper.

DSRC PHY uses OFDM modulation scheme to multiplex data.

Along with the successful deployment of IEEE 802.11a WLAN services and devices in recent years, OFDM has gained increased popularity in the wireless communication community due to its high spectral efficiency, inherent capability to combat multi-path fading and simple transceiver design. In a nutshell, the input data stream is divided into a set of parallel bit streams and each bit stream is then mapped onto a set of overlapping orthogonal sub carriers for data modulation and demodulation. All of the orthogonal sub carriers are transmitted simultaneously. By dividing a wider spectrum into many narrow band sub carriers, a frequency selective fading channel is converted into many flat fading channels over each sub carrier if the sub carrier spacing is small compared to the channel coherence bandwidth. Thus, a simple equalization technique could be used in the receiver to combat the inter-symbol interference. DSRC uses 64 sub carriers where 52 sub carriers are actually used for signal transmission. Out of these 52 sub carriers, 48 are data sub carriers and 4 sub carriers are pilot symbols used for phase tracking. Figure 2.1 shows the training sequence structure both in time and frequency. Two long training symbols are across all the sub-carriers and 4 pilot sub carriers are only embedded in sub carriers -21, -7, 7 and 21.

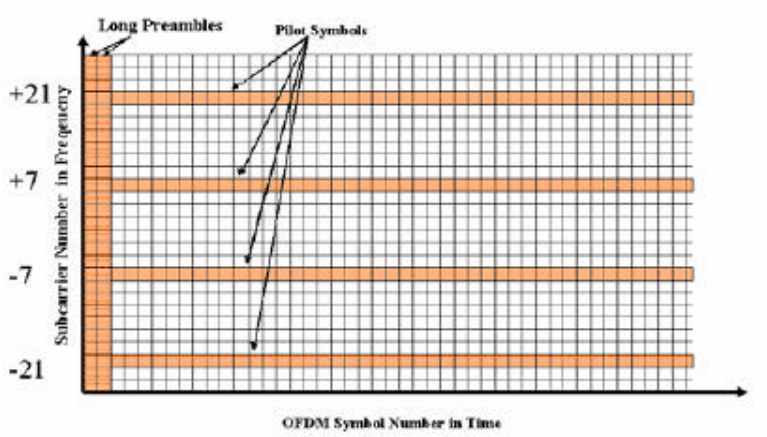


***Figure 2.1**: Two long preambles and pilot sub carriers in time and Frequency*

The following figure shows the physical layer data frame structure.

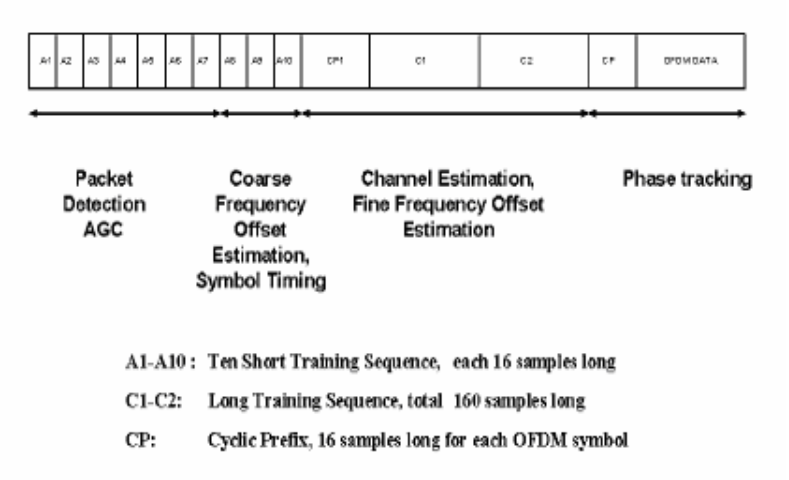


Figure 2.2*. DSRC PHY frame format.*

A1-A10 are ten identical short training symbols, each is 16 samples long.
A subset of these symbols are used for packet detection, automatic gain control (AGC), and various diversity combining schemes. The remaining short training symbols are used for coarse frequency offset estimation and coarse symbol timing estimation. These short training symbols are followed by two long identical training symbols, C1-C2, which are used for channel estimation, fine frequency and symbol timing estimation. C1 and C2 are 64 samples long and the 32-sample long CP1 is the cyclic prefix which protects against inter-symbol interference (ISI) from the short training symbols. After short and long training symbols, comes the actual modulated payload OFDM symbol. The first OFDM data symbol is the physical layer header which is BPSK modulated and specifies the modulation scheme used in the payload OFDM symbols that follows.
Each OFDM symbol consists of 64 samples and a 16-sample long CP which is pre-appended for each OFDM symbol to combat ISI.
Some of the key physical layer parameters used in DSRC are listed in Table I.

Finally, it is worth noting that IEEE 802.11a, from DSRC derives, is primarily designed for indoor WLAN applications. Thus, all PHY parameters are optimized for the indoor low-mobility propagation environment.
Aside from the fact that the DSRC signal bandwidth is 10 MHz (half of the IEEE 802.11a signal bandwidth) in addition to some differences in the transmit power limit, the DSRC PHY follows exactly the same frame structure, modulation scheme and training sequences specified by IEEE 802.11a PHY. However, DSRC applications require reliable communication between On-Board Units (OBUs) and from OBU to Roadside Unit (RSU) when vehicles are moving up to 120 miles/hour and having communication ranges up to 1000 meters. This environment is drastically different from the indoor low-mobility environment and its implications on the DSRC PHY performance turn out to be non-trivial.

Europe has not rapidly made similar decisions as the U.S about intelligent vehicle safety and, at time of this writing, has not allocated a spectrum equivalent to DSRC yet.

Chapter 3

An Overview of GPS, the Global Positioning System

Linking an IP Address with a geographic location has been of interest for quite some time. The first attempt to design a system that actually routes packets according to their geographic destination is *Cartesian Routing* by Gregory G. Finn [8].
The recent redesign of the Internet Protocol (IP) [9] and the advent of the Global Positioning System (GPS) [58, 11, 12, 14] have given a new stimulus for this work. In the proposed redesign of IP, IP address type space was specifically allocated for geographic addresses [10]. IP addresses would be assigned to subnets and hosts based on topological criteria, such as geography [12].

GPS opens new frontiers to several application scenarios.
Above all, MANETs' hosts can exploit GPS devices to acquire their position, thus enabling position based routing and position based applications.
Position is first class information in Ad Hoc scenarios because it can enable mechanisms working on the base of geographic location. This localized approach reflects the localized nature of a MANET network and the need for several applications to work in a defined geographic area.
By connecting to a GPS receiver, a mobile host will be able to know its current physical location. This can greatly help the performance of a MANET, and it is for this reason that many researchers have proposed to adopt GPS in MANETs. For example, mobile hosts in a MANET can avoid using naïve flooding to find routes and neighbours' or destinations' locations may be used as a guideline to find routing paths efficiently.
The applications of location information are not limited to routing protocols.
Location information, when integrated into MANETs, may provide many potential services; some of them are the followings:
• *Navigation:* When location information and wireless communication capability are integrated into navigation systems, users will be able to talk to each other in an ad hoc manner. Navigation systems, which already incorporate GPS, can further combine MANET for group communications. Quick wireless communication links can be established whenever needed. A user will be able to find out who is at what location. Location-dependent emergency rescue and law enforcement services would be possible.

• *Geocast, GeoMulticast:* The goal of Geocast is to send messages to all hosts in a specific area. When urgent events (such as fires, traffic accidents, or natural disasters) occur in a specific area or we want to advertise some information to people in certain areas, Geocasting and Geomulticasting can be convenient ways to achieve this goal.
• *Tour guide:* Tour guide systems can provide location-dependent information to tourists (such as map, traffic, and site information). The effort needed to search for tourism information can be significantly reduced with the help of positioning.

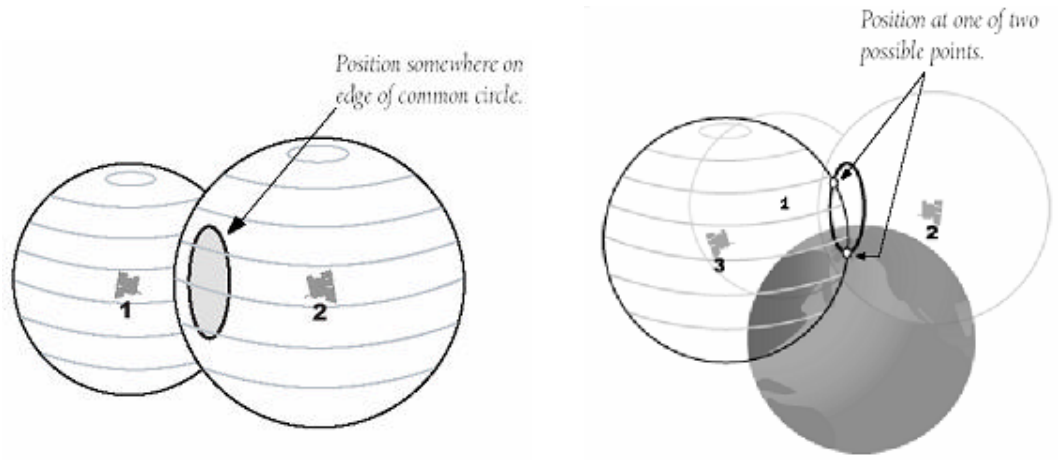
The *Global Positioning System* (GPS) is a satellite-based positioning system that the U.S. government initiated and that is already widely used around the world. GPS is not only used for navigation and positioning (for instance, by ships), but it is also used for synchronization. *cdmaOne* and its migration systems use GPS receivers to synchronize base stations because this method provides high accuracy.
Because people saw the GPS system as a significant advantage in warfare, the U.S. government added a small disturbance to the signals so that the highest accuracy level would only be available to the U.S. Army. This system of *Selective Availability* (SA) was then removed in May 2000 and enabled high accuracy for everyone. Consequently, GPS receivers now enable users to locate themselves with an accuracy of 5m to 40m, depending on the conditions.
It is made up of a "constellation" of 27 Earth-orbiting satellites (24 in operation and three extras in case one fails). This is the initial declared configuration, now they are more than 29.
Each of these 3,000- to 4,000-pound solar-powered satellites circles the globe at about 12,000 miles (19,300 km), making two complete rotations every day.
The orbits are arranged so that at any time, anywhere on Earth, there are at least four satellites "visible" in the sky.
A GPS receiver's job is to locate four or more of these satellites, figure out the distance to each, and use this information to deduce its own location. This operation is based on a simple mathematical principle called *trilateration*. In order to perform this calculation, the GPS receiver has to know two things:

1. The location of at least three satellites above it.

2. The distance between it and each of those satellites.

Three satellites are needed to perform a trilateration and find a position on the earth, but receivers generally look to four or more satellites, however, to improve accuracy and provide precise altitude information.
If we know we are *X* miles from satellite A in the sky, we could be anywhere on the

surface of a huge, imaginary sphere with an *X*-miles radius. If we also know we are *Y* miles from satellite B, we can overlap the first sphere with another sphere.
The spheres intersect in a perfect circle.
If we know the distance to a third satellite, we get a third sphere, which intersects with this circle at two points.
The earth itself can act as a fourth sphere -- only one of the two possible points will actually be on the surface of the planet, so we can eliminate the one in space.
A GPS receiver figures out both locations of satellites and distances between it and each of those satellites by analyzing high-frequency, low-power radio signals from the GPS satellites. Radio waves are electromagnetic energy, which means they travel at the speed of light (about 186,000 miles per second, 300,000 km per second in a vacuum). The receiver can figure out how far the signal has travelled by timing how long it took the signal to arrive.
Better units have multiple receivers, so they can pick up signals from several satellites simultaneously.



a. b.

Figure 3.1. *To locate itself, a GPS receiver must find the distance to three satellites of known positions. If the receiver finds that it is x miles from one satellite, it knows that it must be somewhere on an imaginary sphere, with the satellite as the centre and a radius of x. Figure* ***a*** *shows that if the receiver can generate these spheres for two satellites, it knows it can only be located where the surfaces of the two spheres intersect. The two spheres overlap in a ring of possible receiver positions. Figure* ***b*** *shows that by generating a sphere for a third satellite, the receiver narrows its possible positions down to two points. Earth itself acts as a fourth sphere and allows the selection of the right point: the receiver dismisses the point located in space, leaving only one possible position.*

Civilian GPS receivers have potential position errors due to the result of the accumulated errors due primarily to some of the following sources [58]:

- *Ionosphere and troposphere delays* – The satellite signal slows as it passes through the atmosphere. The system uses a built-in "model" that calculates an average, but not exact, amount of delay.

- *Signal multi-path* – Occurs when GPS signal is reflected off objects such as tall buildings or large rock surfaces before it reaches the receiver. This increases the travel time of the signal, thereby causing errors.

- *Receiver clock errors* – Since it is not practical to have an atomic clock in a GPS receiver, like satellites have, the built-in clock can have very slight timing errors.

- *Orbital errors* – Also known as "ephemeris errors", these are inaccuracies of the satellite's reported location.

- *Number of satellites visible* – The more satellites the receiver can "see", the better the accuracy. Buildings, terrain, electronic interference, or sometimes even dense foliage can block signal reception, causing position errors or possibly no position reading at all. The clearer the view, the better the reception. GPS units will not work indoors (typically), underwater or underground.

- *Satellite geometry/shading* – This refers to the relative position of the satellites at any given time. Ideal satellite geometry exists when the satellites are located at wide angles relative to each other. Poor geometry results when the satellites are located in a line or in a tight grouping.

- *Intentional degradation of the satellite signal* – The U.S. military's intentional degradation of the signal is known as "Selective Availability" (SA) and is intended to prevent military adversaries from using the high accurate GPS signals. SA accounts for the majority of the error in the range. SA was turned off May 2, 2000 and is currently not active.

Since GPS does not provide exact location information, the Differential GPS (DGPS) system [58, 14] has been introduced to provide more accurate location information. In the basic form of DGPS, a reference station with a precisely known location is used. DGPS works by placing GPS receivers (called *reference receivers*) at known locations. Since a reference receiver knows its exact location, it can determine errors in the satellite signals. By comparing the result of location information obtained by GPS signal with the pre-known location information, the reference receiver can produce error correction information. The error correction information is broadcasted by the DGPS

transmitter to DGPS receivers and used for error correction of signals from the satellites. Some DGPSs can provide exact location information with no more than 1 m error.

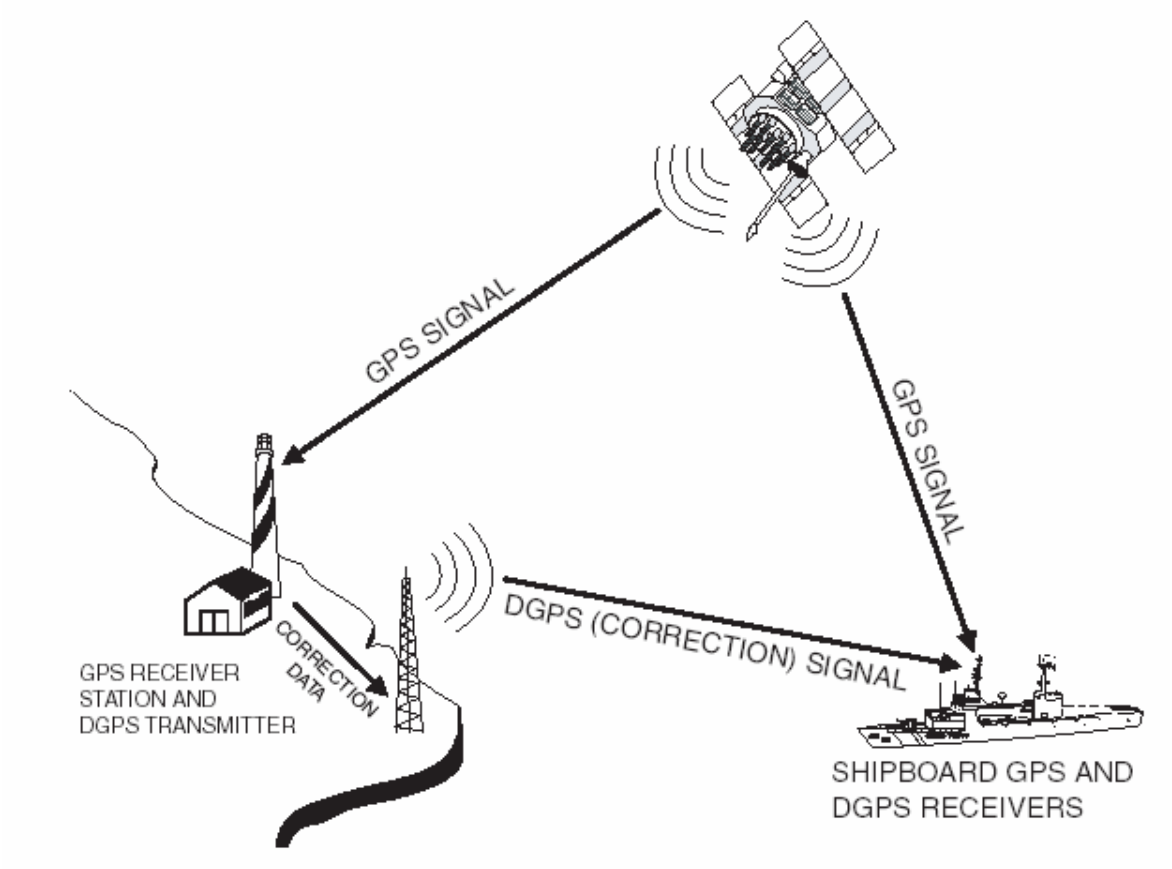


Figure 3.2. *DGPS architecture*.

The satellites never need to consider how many users they are serving and where those users are; rather, they just broadcast the signal for the receivers to pick up. As the name indicates, the GPS receiver does not have to send anything to the satellites, which means less power consumption. Still, the computations that the GPS receiver needs in order to perform are significant if the processing takes place in the device (an important consideration for device manufacturers). Some manufacturers choose to integrate a GPS chip into the mobile device while others use a dedicated GPS receiver and a Bluetooth

or cable connection in order to attach it. The use of GPS chips has started to pick up as size, power consumption, and price have gone down and paved the way for a more widespread use of this technology.
The most significant limitation of GPS has always been that that it requires a clear view of the sky. Consequently, it is likely that a mobile device that a user is using in a car will only be capable of using GPS if the antenna is located outside the car, which might reduce the device's usability. Car manufacturers are therefore looking into solutions where the mobile device is built into the car and attached to an external antenna. Another solution is to build the GPS antenna into the car and let whatever device the user has connect to it by using Bluetooth. Likely, GPS positioning solutions will also be complemented with a network-based solution, such as the cell identity, that does not need line-of-sight to the satellites.

Network-Assisted GPS (A-GPS) [58] uses network-based GPS receivers to help the terminal measure GPS data. These receivers are placed around the mobile network in 200 to 400 km intervals and collect GPS satellite data on a regular basis. These can then be requested from the GPS-enabled terminal and enable the receiver to make timing measurements without having to decode the actual messages from the satellites. This process reduces the TTFF to one to eight seconds and makes GPS a much more compelling positioning solution.

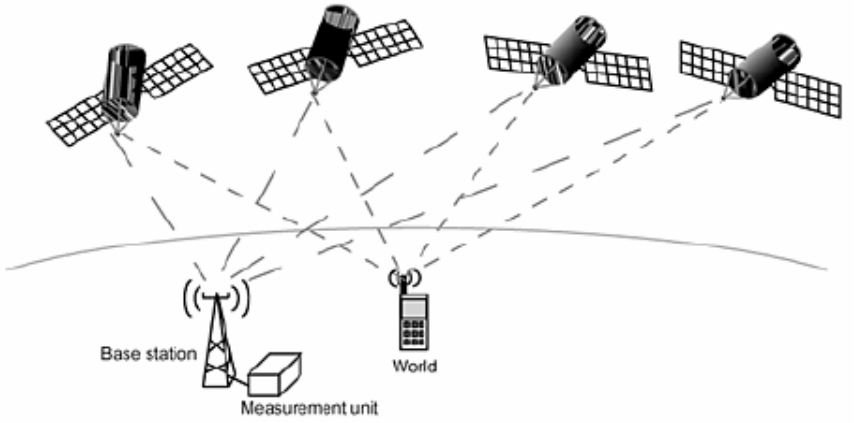


Figure 3.3. *A-GPS architecture*

An interesting feature of GPS-based positioning solutions is that they enable user locating in three dimensions. For some specific applications, such as 911 rescue operations in the mountains, this feature might be of value because the rescuers can immediately see at what height the user is located.

Finally, some functions need positioning that is accurate to within centimetres. Those functions require the new generation of GPS satellites or the planned European GALILEO satellite network that will send out directional signals. These will enable systems to determine the position of objects to within a few centimetres using the carrier differential GPS method (CDGPS).
The CDGPS method calculates the number of directional signal wavelengths between the satellite and the receiver. A wavelength of 20 centimetres and a phase measurement accuracy of five degrees yields a positioning accuracy in centimetres.
Furthermore, new CDGPS signals can measure a vehicle's position every second and this allows accurate and "real" real-time position tracking.

Chapter 4

Position Based Routing

1. Position Based Routing

"In the near future, Global Positioning System (GPS) cards will be deployed in each car and possibly in every user terminal. A user's *location* will become information that is as common as the *date* is today, getting input from GPS, when outdoors, and other location providing devices, when indoors. Availability of location information will have a broad impact on application-level as well as on network-level software. "[9].

In mobile ad hoc networks, systems may move arbitrarily.
Examples where mobile ad hoc networks may be employed are the establishment of connectivity among handheld devices or between vehicles. Since mobile ad hoc networks change their topology frequently and without prior notice, routing in such networks is a challenging task. At large extent, we can distinguish two different approaches: *topology-based* and *position-based* routing.

Topology-based routing protocols use the information about the links that exist in the network to perform packet forwarding.
They can be further divided into *proactive (table-driven)*, *reactive (on-demand)* and *hybrid* approaches.

The increasing size and use of wireless communication systems strengthens the need for scalable wireless routing algorithms. Standard Internet routing achieves scalability through address aggregation, in which each route announcement describes route information for many nodes simultaneously. This approach to scalability is not applicable to many wireless environments, such as ad hoc networks or sensor-nets, where the node identifiers of topologically and/or geographically close nodes may not be similar (*e.g.*, by sharing high-order bits). For these cases, two main scalable routing techniques have been proposed: on-demand (reactive) routing and geographic (position-based) routing.
Thus, *reactive* and *position based* routing algorithms give, at a large extent, better results than other routing protocols in ad-hoc environments.

Position-based routing algorithms eliminate some of the limitations of topology-based routing by using additional information.
They require that information about the physical position of the participating nodes be available.
Position-based algorithms avoid overhead by requiring only accurate neighbourhood information and a rough idea of the position of the destination.

They are *localized algorithms* (and so distributed in nature) and often exploit greedy techniques, where simple local behaviour (try to) achieves a desired global objective [11, 12, 16, 17, 18, 19, 40, 41, 42, 44, 45, 47].

Commonly, each node determines its own position through the use of GPS or some other types of positioning service [12, 13, 14, 15].
In a unicast scenario, a *location service* is needed by the sender of a packet to determine the position of the destination and to include it in the packet's destination address.
The routing decision at each node is then based on the destination's position contained in the packet and the position of the forwarding node's neighbours.
Position-based routing thus does not require the establishment or maintenance of routes.
The nodes have neither to store routing tables nor to transmit messages to keep routing tables up to date. As a further advantage, *position-based routing supports the delivery of packets to all nodes in a given geographic region in a natural way*. This type of service is called *Geocasting* [24, 27, 28, 29, 30, 31, 32, 33, 35, 36].

The general idea of position-based routing is to select the next hop of a packet based on position information such that the packet is forwarded in the geographic direction of the destination. This forwarding decision is based purely on local knowledge. It is not necessary to create and maintain a "global" route from the sender to the destination. Therefore, *position-based routing is commonly regarded as highly scalable and very robust against frequent topological changes* and it is believed to be suitable to vehicular networks due to their characteristics:
- a very large number of nodes with a relatively high density
- a very general communication pattern with many host pairs communicating
- a need for low latency first-packet delivery
- a highly mobile topology

A pre-requisite for position-based unicast routing is to know the position of the destination of a packet. For this purpose, distributed algorithms called *location services* have been proposed
Thus, with unicast geographic routing, the task of routing packets from a source to a destination can be separated into two distinct aspects:
• Discovering the position of the destination
• The actual forwarding of packets based on this information.

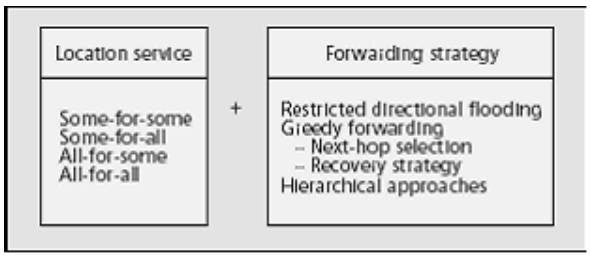


Figure 4.1. Building blocks and criteria for classification.

2. Location Services

Before a packet can be sent, it is necessary to determine the position of its destination. Typically, a *location service* is responsible for this task. Existing location services can be classified according to how many nodes host the service. This can be either *some* specific nodes or *all* nodes of the network.
Furthermore, each location server may maintain the position of *some* specific or *all* nodes in the network.
The four possible combinations can be abbreviated as: some-for-some, some-for-all, all-for some, and all-for-all.

In order to learn the current position of a specific node the help of a *location service* is needed. Mobile nodes register their current position with the service. When a node does not know the position of a desired communication partner, it contacts the location service and requests that information. In classic cellular networks, there are dedicated position servers (with well-known addresses) that maintain position information about the nodes in the network. With respect to the classification, this is a some-for-all approach since the servers are *some* specific nodes, each maintaining position information about *all* mobile nodes.
In mobile ad hoc networks, such a centralized approach is viable only as an external service that can be reached via non ad-hoc means. There are two main reasons for this. First, it would be difficult to obtain the coordinates of a position server if the server were part of the ad hoc network itself. This would represent a chicken-and-egg problem: without a position server, it is not possible to get position information, but without the position information the server cannot be reached. Second, since an ad hoc network is

dynamic, it might be difficult to guarantee that at least one position server will be present in a given ad hoc network.
The best approach seems to be a decentralized location service which is part of the ad hoc network.
The following described algorithms are of that kind and are *proactive* algorithms.

2.1 DREAM - Distance Routing Effect Algorithm for Mobility - Location Service

Distance Routing Effect Algorithm for Mobility is a protocol which defines a location service and a forwarding paradigm, too. Within the DREAM framework [22], each node maintains a position database that stores position information about each other node that is part of the network. It can therefore be classified as an all-for-all approach. An entry in the position database includes an identifier for the node, the direction of and distance to the node, as well as a time value that indicates when this information was generated. Of course, the accuracy of such an entry depends on its age.
DREAM was built around two observations.
One, called the *distance effect*, uses the fact that the greater the distance separating two nodes, the slower they appear to be moving with respect to each other. Accordingly, the location information in routing tables can be updated as a function of the distance separating nodes without compromising the routing accuracy.
The second idea is that of triggering the sending of location updates by the moving nodes autonomously, based only on a *node's mobility rate*. Thus in a directional routing algorithm, routing information about the slower moving nodes needs to be updated less frequently than that about highly mobile nodes. In this way each node can optimize the frequency at which it sends updates to the networks and correspondingly reduce the bandwidth and energy used, leading to a fully distributed and self-optimizing system.

Each node regularly floods packets to update the position information maintained by the other nodes. A node can control the accuracy of its position information available to other nodes by:
• the frequency at which it sends position updates (*temporal resolution*);
• indicating how far a position update may travel before it is discarded (*spatial resolution*)

The temporal resolution of sending updates is coupled with the mobility rate of a node (i.e., the higher the speed, the more frequent the updates). The spatial resolution is used

to provide accurate position information in the direct neighbourhood of a node and less accurate information at nodes farther away.

DREAM's Location Service could be a valid choice for inter-vehicles communications due to its low overhead in the query phase because it is an all-to-all approach; but this comes with its burden because building tables in every node for all other nodes introduce considerable overhead.

2.2 Quorum based Location service

The *Quorum-based scheme* [Haas and Liang, 1999] intends to provide location service using the concept of quorum that is widely used in distributed database design. A number of hosts are designated as the location service providers. These hosts are partitioned into a number of quorum sets $Q_1, Q_2, ..., Q_k$. The design of quorums should guarantee that for each $i \geq 1, j \leq k$,

$$Q_i \cap Q_j \neq 0$$

When a host changes its location, it can pick any nearest quorum Q_i to update its location (based on any optimization criteria). When a host needs any other host's current location, it can query any nearest quorum Q_j. Since the intersection of Q_i and Q_j must be non-empty, the most up-to-date location information can be obtained.

The Quorum-based position service can be configured to operate as all-for-all, all-for-some or some-for-some approaches, depending on how the size of the backbone and the quorum is chosen. However it typically works as a some-for-some scheme with the backbone being a small subset of all available nodes and a quorum being a small subset of the backbone area.

2.3 GLS - Grid Location Service

The *Grid Location Service* (GLS) [52, 15] protocol is a decentralized routing protocol. Each mobile node periodically updates a small set of other nodes (its *location servers*) with its current location. A node sends its position updates to its location servers without knowing their actual identities, assisted by a predefined ordering of node

identifiers and a predefined hierarchy. Queries for a mobile node's location also use the predefined ordering and spatial hierarchy to find a location server for that node. For example, when node *A* wants to find the location of node *B*, it sends a request to the least node greater than or equal to node *B* for which it has location information. That node forwards the query in the same way, and so on. Eventually, the query will reach a location server of node *B*, which will then forward the query to node *B*. Since the query contains node *A*'s location, it can respond directly using geographic forwarding.
Routing updates are carried out using either flooding based algorithm or link reversal algorithm.

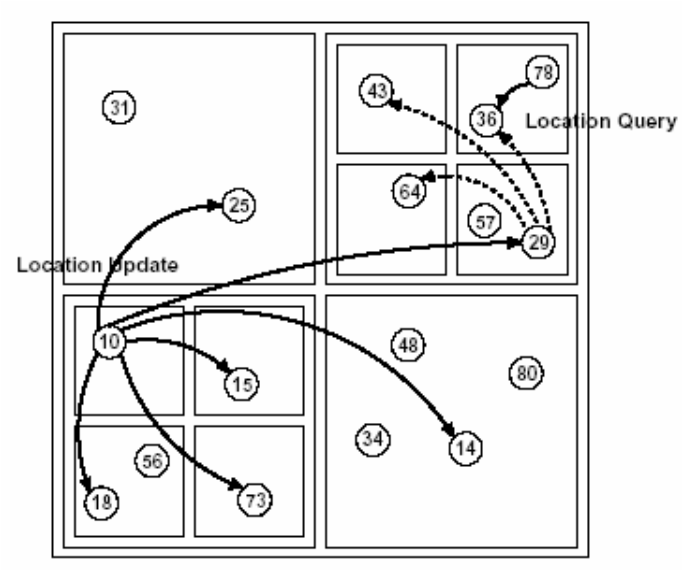


Figure 4.2*. An example of GLS.*

The Grid Location Service (GLS) divides the area covered by the ad hoc network into a hierarchy of squares. In this hierarchy, n-order squares contain exactly four (n - 1) order squares, constructing a so-called quad tree. The lowest order squares typically have a size comparable to the radio range of a node whereas the highest order square covers the whole network. Each node maintains a table of all other nodes within the local first-order square. The table is constructed with the help of periodic position broadcasts, which are scoped to the area of the first-order square.
To determine where to store position information, GLS establishes a notion of near node IDs, defined to be the least IDs greater than a node's own ID within a given n-order square. (ID numbers wrap around after the highest possible ID.)
When a node A wants to distribute its position information, it sends position updates to

the node with the nearest IDs in each of the three surrounding first-order squares. Thus, the position information is available at 3 nodes and all nodes that are in the same first-order square as the sending node itself. In the surrounding three second-order squares, again the nodes with the nearest IDs (to the first node A) are chosen to host the node's position. This process is repeated until the area of the ad hoc network is covered. The "density" of the position information decreases logarithmically with the distance to the node that updates its position.
A node does not need to know the IDs of its position servers, which makes a bootstrapping mechanism that discovers a node's position servers unnecessary. Position information is forwarded to nodes with nearer IDs in a process closely resembling position queries, only that information is written instead of read, ensuring that the position information reaches the correct node, where it is then stored.
Since GLS requires that all nodes store the information of some other nodes, it can be regarded as an *all-for-some* approach. The burden of maintaining position information is distributed evenly among all nodes. The hierarchical structure of GLS allows scaling to large networks, since update and query complexities both scale with $O(\sqrt{n})$.

2.4 Homezone

Another distributed way to store location information is the *Homezone* mechanism.
The Homezone [Giordano and Hamdi 1999, 52] and Home Agent location services introduce the concept of a virtual home region of a node. All nodes within the virtual home region of a certain node have to maintain up-to-date position information for that node. Through a well-known hash function, the identifier of a node is hashed to a position, and the virtual home region is formed by all the nodes within a certain radius of that position. The radius has to be chosen such that the virtual home region contains a sufficient number of nodes. To obtain the position of a node, the same hash function is applied to the node identifier, and a location query can then be sent to the resulting position of the home region. Any node within the home region can answer the query.
The location database is distributed among all hosts but the schemes operate on a flat set of nodes without any hierarchy. The reduction in complexity comes at the expense of flexibility and efficiency. Nodes can be hashed to a far away home region leading to high response delays. Furthermore, if only one home region per node exists, it is possible that the home region of a node cannot be reached (e.g., because of network partitioning). If the home region is sparsely populated, its size has to be increased. As a consequence, several tries with increasing radius around the centre of the home region may be necessary for location update messages as well as queries.

When a host needs other hosts' locations, it queries their homezones by computing the hashing function. This solution is classified as an *all-for-some* approach.
Since only the home region is queried and updated, the overall communication overhead of this scheme scales with $O(\sqrt{n})$.

2.5 GRSS - Geographic Region Summary Service

Geographic Region Summary Service (GRSS) [52], like GLS, uses a grid-like hierarchy of the network. In this location service, the network is logically partitioned into overlapping squares of different orders, as described in the previous section on GLS. This structure is a priori known such that each node knows in which square of each order it resides.
All nodes know their neighbours within the smallest square (order-0). A local routing protocol (using link-state routing) generates exact paths to each of them. The size of order-0 squares should be chosen small enough so that the local link-state routing protocols provide valid routes even when node mobility is high (e.g., two-hop radio range).
Using the knowledge of all node IDs residing in the same order-0 square, the nodes at the border of an order-0 square periodically generate a summary. This summary is a bit vector with one bit for each node ID in the network. A bit is set if the corresponding node is in the order-0 square, otherwise it is not set. The summary is transmitted to the adjacent order-0 square, where the information is flooded.
Thus the nodes in an order-0 square know which nodes are located in the neighbouring order-0 squares.
The nodes at the border of the order-1 square generate a summary of all nodes in that order-1 square and transmit this information to the neighbouring order-1 squares, repeating the process described above.
A boundary node of order n (i.e., the shared order of the boundary node and its siblings) generates summaries for each of the squares it is located in up to the order of n.
At the end of this process, a union of all summaries covers the whole network except for the own order-0 square, for which complete routing information is available. Similar to DREAM, the closer a node is located to a given node, the more precise the knowledge of its position gets. For example, if two nodes are located in the same order-0 square, they know the exact position of each other. If they are located in different order-0 squares but in the same order-1 square they know in which order-0 square the other node resides and so on.

Besides the "exact summary" generation that uses one bit for each node in the network, GRSS also supports "imprecise summary" generation. In order to decrease the length of the bit vector, imprecise summary generation uses the technique of *bloom* filters [52]. To generate an imprecise summary, t hash functions (for t >= 1), which generate t bit positions in the vector, are applied to the node's ID. If a node resides in the generating node's own order-0 square, all bits at the t positions are set in the vector. If one wants to know if a certain node (ID) is located in the corresponding square, the t hash functions with the node's ID as an input are applied to the bit vector. If all the corresponding bits are set, the node is located in that square with a very high probability. However, there is the possibility of a set of different nodes setting the bits belonging to the node ID ("false positive"). The consequence of a false positive is that there are two or more alternatives for the location of the node. Thus, one would have to either duplicate the packet, resulting in additional overhead, or just choose one alternative, which can result in a detour of the packet. Typically one would choose the latter option to remain scalable.
As all nodes have some information about the location of all other nodes, GRSS is an all-for-all approach. When used with precise summary generation, GRSS has similar characteristics to DREAM: position updates scale with O(n) and position lookups are free. The imprecise summary generation may be used to reduce the overhead of position updates at the cost of collisions.

2.6 RLS - Reactive Location Service

Unlike the aforementioned Location Service algorithms, RLS [44, 45] is an on-demand approach to the same problem.
RLS is inspired by DSR route discovery: whenever the position of a node is required, the node looking for position information floods a request containing the ID of the node it is looking for.
The request contains the ID and position of the requesting node. When a node receives a request with its own ID, it replies to the node looking for its position.
RLS stores information about the location of a node only at the node itself. Querying the location of a node is equivalent to reaching that node with the query, and the node can then respond with its location. A node's position is queried by flooding the query packet. Instead of immediately flooding with the maximum hop distance (i.e., the diameter of the network), it is possible to gradually increment the flood radius until the corresponding node is reached. The characteristics of the algorithm are largely determined by the chosen method of incrementing the search radius (e.g., linear or exponential) and the time intervals between successive attempts.

RLS does not require any position updates. The overhead of a single position query scales with O(n).
The mechanism is fairly simple to implement and very robust against node failure or packet loss.
Furthermore, the location service only consumes resources when data packets have to be sent. Since only the node itself maintains its location, storage requirements are minimal. Nevertheless, the overhead caused by flooding location requests makes such a mechanism unsuitable for scenarios where location queries are frequent or the network is large.

3. Forwarding Strategies

In position-based routing, the forwarding decision by a node is primarily based on the position of a packet's destination and the position of the node's immediate one-hop neighbours.
The position of the destination is contained in the header of the packet. If a node happens to know a more accurate position of the destination, it may choose to update the position in the packet before forwarding it. The position of the neighbours is typically learned through one-hop broadcasts.
These beacons are sent periodically by all nodes and contain the position of the sending node. However solutions like CBF (Contention Based Forwarding) [23] could be used, instead.
This kind of solution is briefly explained here after the greedy forwarding solution.

We can distinguish three main packet forwarding strategies for position-based routing:

- Greedy forwarding
- Restricted directional flooding
- Hierarchical approaches

With the first two approaches, a node forwards a given packet to one (greedy forwarding) or more (restricted directional flooding) one-hop neighbours that are located closer to the destination than the forwarding node itself. The selection of the neighbour in the greedy case depends on the optimization criteria of the algorithm. There are diverse strategies that existing algorithms use to make this selection.
Examples of greedy algorithms are GEDIR, MFR and Random Progress Method.
It is fairly obvious that both forwarding strategies may fail if there is no one-hop neighbour that is closer to the destination than the forwarding node itself. *Recovery strategies* that cope with this kind of failure try to escape from local optimum and find a way forwarding can carry on. FACE algorithms [17] are a typical example of a recovery strategy algorithm.
The algorithms presented in [17] have been improved by other studies like [18].
A classical protocol which uses a combination of greedy forwarding and a recovery strategy is Greedy-Face-Greedy (GFG); another is Greedy Perimeter Stateless Routing (GPSR), an implementation of GFG.
DREAM and LAR are restricted directional flooding algorithms.

The third forwarding strategy consists in constructing a hierarchy of mobile nodes in order to scale to a large number of nodes. Two representatives of hierarchical routing that use greedy forwarding for wide area routing and non-position based approaches for

local area routing are ZRP and GRID [52, 15].

3.1 Greedy Forwarding

In a localized routing scheme like geographic routing is, a node *S*, currently holding a message, is aware only of the position of its neighbours within its transmission radius and the destination *D*.

Takagi and Kleinrock [16, 33] proposed the first position-based routing scheme, based on the notion of progress. Given a transmitting node *S*, the *progress* of a node *A* is defined as the projection onto the line connecting *S* and *D*. In the *MFR (Most Forward within Radius)* scheme [33], the packet is forwarded to a neighbour with maximal progress. Nelson and Kleinrock also discussed the *random* progress method (choosing at random one of the nodes with progress and adjusting the transmission radius to reach up to that node), arguing that there is a trade-off between the progress and transmission success.

Finn [16] proposed the greedy routing scheme based on geographic distance: the source *S* selects neighbour node *G* that is closest to the destination among its neighbours. Only neighbours closer to the destination than *S* are considered. Otherwise there is a lack of advance, and the method fails.
A variant of this method is called the *GEDIR* [15, 16].

3.1.1 GEDIR - Geographic Distance Routing

The *geographic distance routing (GEDIR)* protocol [Lin and Stojmenovic, 1999] assumes that each host has the locations of its direct neighbours.
All neighbours are considered, and the message is dropped if the best choice for a current node is to return the message to the node the message came from (stoppage criterion indicating lack of advance).

GEDIR protocol directly forwards packets to next hops without establishing routes in advance.
There are two packet-forwarding policies:

- *distance approach* and
- *direction approach*.

In the distance approach, the packet is forwarded to the neighbour whose distance is nearest to the destination.

In the direction approach, the packet is forwarded to the neighbour whose direction is closest to the destination's direction.
The latter can be formulated by the angle formed by the vector from the current host to the destination and to the next hop.
The distance approach may lead a packet to a local maximum host, while the direction approach may lead a packet into an endless loop. In order to resolve these problems, several variations are proposed, such as the f-GEDIR ("f" stands for flooding) and *c*-GEDIR (i.e., concurrently sending from the source to *c* hosts). These mechanisms are used to help the packet leave the local maximum host or the loop.
To further improve the performance of GEDIR, [Stojmenovic and Lin, 2001] recommends that hosts collect the locations of their two-hop neighbours. A host, on requiring to send/forward a packet, first picks a host (say *A*) from its one/two-hop neighbours whose distance (or direction) is nearest (or closest) to the destination.
If host *A* is a one-hop neighbour, the packet is directly forwarded to *A*. Otherwise, the packet is forwarded to the host that is *A*'s one-hop neighbour. The protocol is called *2-hop GEDIR*. This protocol can also be combined with flooding to discover a route. Both GEDIR and 2-hop GEDIR have been proven to be loop free.

3.1.2 About beacon broadcasts in greedy forwarding strategies

In all existing strategies for greedy unicast forwarding, the position of a node is made available to its direct neighbours (i.e., nodes within single-hop transmission range) in form of periodically transmitted beacons (e.g. GEDIR for GFG, GPSR's greedy part). Each node stores the information it receives about its neighbours in a table and thus maintains position information about all direct neighbours. The state expires after a certain amount of time.
Given its own position, the "last-known" position of the direct neighbours, and the position of the destination of the packet, a node selects a next hop out of his neighbours table according to a forwarding strategy. One frequently used heuristic is picking the neighbour minimizing the remaining distance to the destination under the constraint that the neighbour has a smaller distance than the forwarding node. Once a neighbour is selected, it is addressed directly with its MAC address. This process is called 'greedy forwarding'.
Greedy forwarding faces the following drawbacks:

1) The position information of the neighbours looses accuracy over time in the presence of mobility. In the worst case a node that was reachable has moved out of range. Since

in general radio links have a high error rate, this situation is often difficult to identify. Usually it is accomplished by assuming that a node is unreachable if it did not acknowledge a packet after a certain number of retries. Thus, the inaccuracy of the neighbour information places additional load on the MAC layer and - if the routing algorithm can not react to this transmission failure - the packet is lost. If the algorithm reacts to the link failure by removing the faulty neighbour from the neighbour table and selecting another neighbour for forwarding, it avoids packet loss at the cost of additional packet delay (and the risk of choosing another unreachable node as forwarder).
2) The beacons themselves impose additional load on the network. The higher the frequency of beacons, the lower the aforementioned inaccuracy of neighbour information but the higher the load on the network.
3) Since beacons are transmitted with link-layer broadcast, a transmission failure can not be detected, resulting in nodes being close and not recognized as being neighbours. This can lead to suboptimal forwarding decisions, the unnecessary initiation of the recovery procedure, or to packet loss. With a high node density, even increasing the beaconing frequency does not help much since the probability of beacon collisions increases as well.
4) The assumption of bi-directional links needed by neighbour-table-based forwarding is not necessarily true for real radio links.

A proposed mechanism to cope with these problems and improve greedy forwarding is Contention Based Forwarding (CBF) [23].
CBF performs greedy forwarding without the help of beacons and without the maintenance of information about the direct neighbours of a node. Instead, all suitable neighbours of the forwarding node participate in the next hop selection process and the forwarding decision is based on the actual position of the nodes at the time a packet is forwarded. This is in contrast to existing greedy forwarding algorithms that base their decision on the positions of the neighbours as perceived by the forwarding node and eliminates the problems outlined above. In order to escape from local optima, existing recovery strategies can either be used directly or may be adapted to be used with CBF.
The contention process of CBF used for next-hop selection represents a paradigm change in the forwarding of packets. In traditional protocols, the forwarder actively selects the desired next-hop by unicasting the packet to the corresponding MAC address. In contrast, with CBF the responsibility for next-hop selection lies with the set of possible next hops.
CBF consists of two parts: the *selection* of the next hop is performed by means of contention, while *suppression* is used to reduce the chance of accidentally selecting more than one next hop. In [23] the authors present three suppression strategies with different characteristics and state that, in their results, the suppression of duplicate

packets works well, that CBF achieves similar packet delivery ratios as beacon-based greedy routing, and that it reduces the load on the wireless medium for a given delivery rate if node mobility is high.
CBF, therefore, could represent a good alternative to traditional beacon-based greedy forwarding.

The general idea of CBF is to base the forwarding decision on the current neighbourhood as it exists in reality and not as perceived by the forwarding node. This requires that all suitable neighbours of the forwarding node are involved in the selection of the next hop.
CBF works in the following three steps.
First, the forwarding node transmits the packet as a single-hop broadcast to all neighbours.
Second, the neighbours compete with each other for the "right" to forward the packet. During this *contention period*, a node determines how well it is suited as a next hop for the packet.
Third, the node that wins the contention *suppresses* the other nodes and thus establishes itself as the next forwarding node.

3.2 Restricted Directional Flooding

In directed flooding, packet duplication does not occur by accident but is part of the standard forwarding algorithm. A node will forward a packet to all neighbours that are located in the direction of the destination.
Directed flooding is very robust at the cost of heavy network load.

3.2.1 DREAM – Distance Routing Effect Algorithm for Mobility

In DREAM [22], the direction toward the destination is determined by means of a so-called expected region.
The expected region is a circle around the position of D as it is known to a forwarding node N. Since this position information may be outdated, the radius *r* of the expected region is set to (t1- t0)**vmax*, where *t1* is the current time, *t0* is the timestamp of the position information that N has about D, and *vmax* is the maximum speed with which a node may travel in the ad hoc network.

Given the expected region, the "direction towards D" can be defined by the line between N and D and the angle. The neighbouring hops repeat this procedure using their information on D's position. If a node does not have a one-hop neighbour in that direction, a recovery strategy is necessary. This procedure is not part of the DREAM specification.
Even though directed flooding limits the flooding to the direction toward the destination, the communication complexity has the same order as pure flooding, O(n); it is just better by a constant factor.
Directed flooding therefore does not scale to large networks with a high volume of data transmissions.
On the other hand, it is highly robust against the failure of individual nodes and position inaccuracy, and it is very simple to implement. This qualifies it for applications that require a high reliability and fast message delivery for very infrequent data transmissions. DREAM works best in combination with an all-for-all location service that provides more accurate information close to the destination. This reduces the size of the expected region and thus the area in which the packet is flooded.

3.2.2 LAR - Location Aided Routing

The *location-aided routing (LAR)* protocol [Ko and Vaidya 1998, 52] assumes that the source host *S* knows the recent location and roaming speed of the destination host *D*. Suppose that *S* obtains *D* 's location, denoted as (*Xd, Yd*), and speed, denoted as *v*, at time *t0* and that the current time is *t*.
We can define the *expected zone* in which host *D* may be located at time *t1*. The radius of the expected zone is $R = v(t1 - t0)$.
From the expected zone, we can define the *request zone*. The LAR protocol basically uses restricted flooding to discover routes. That is, only hosts in the request zone will help forward route-searching packets. Thus, the searching cost can be decreased. When *S* initiates the route-searching packet, it should include the coordinates of the request zone in the packet. A receiving host simply needs to compare its own location to the request zone to decide whether or not to rebroadcast the route-searching packet.
After *D* receives the route-searching packet, it sends a route reply packet to *S*.
When *S* receives the reply, the route is established. If the route cannot be discovered in a suitable timeout period, *S* can initiate a new route discovery with an expanded request zone. The expanded request zone should be larger than the previous request zone. In the extreme case, it can be set as the entire network. Since the expanded request zone is larger, the probability of discovering a route is increased with a gradually increasing cost.

3.3 Hierarchical Approaches

The introduction of a hierarchy is a well-known method in traditional networks to reduce the complexity each node has to handle. It allows networks to scale to a very large number of nodes. Currently there exist two approaches that introduce a two-layer hierarchy to routing in ad hoc wireless networks: *Terminodes Routing* and *Grid Routing*. Both approaches combine the use of a non-position-based approach on one level of the hierarchy with a position-based approach at the other level.

3.3.1 GRID

With GRID [52, 15], the geographic area is partitioned into squares called *grids*. In each non-empty grid, one mobile host is elected as the *leader* of the grid.
Routing is then performed in a grid-by-grid manner. Only the grid leaders have the responsibility to relay data packets.
Location information is utilized in GRID in this way:

• *Route Discovery:* The concept of the request zone, similar to that in LAR [Ko and Vaidya, 1998], is used to confine the route-searching area. In addition, only grid leaders are responsible for forwarding route-searching packets. Non-leaders' route-searching packets are likely to be redundant since hosts in the same grid are close to each other (and so are their neighbours).
Therefore, GRID can significantly save route-searching packets.

• *Packet Relay:* In GRID, a route is not denoted by host ID. Instead, it is denoted by a sequence of grid ID's. Each entry in a routing table records the next grid leading to a particular destination.
This provides an interesting "handoff" capability in the sense that if a host roams away, the next leader (if any) in the same grid can take over and serve as the relay host without breaking the original route. Thus, GRID has been shown to be more resilient to host mobility.

• *Route Maintenance:* In GRID, routes are maintained by re-electing a new leader if the previous leader moves away. Therefore, the route is still alive. On the contrary, in most other protocols, such as DSR, AODV, LAR, and ZRP [52], once any intermediate host in a route roams away, the route is considered broken. Further, even if the source *S* roams into another grid, the route may still remain alive.
In each grid, hosts have to run a leader election protocol to maintain its leader. When a

leader roams off its original grid, a “handoff” procedure needs to be executed to pass its routing table to the newly elected leader. In most other routing protocols, such a handover procedure is not possible.
Thus, routes in GRID can survive for a longer lifetime. Therefore, GRID is less vulnerable than most other routing protocols to host mobility.
In addition, the amount of control traffic is quite insensitive to host density.
These merits make GRID quite scalable.

3.4 GFG, GPSR and GOAFR+

The most widely used approach to perform geographic routing utilizes the greedy approach complemented with a recovery strategy where greedy forwarding fails.
Geographic routing uses nodes’ locations as their addresses and forwards packets (when possible) in a greedy manner towards the destination. The most widely known proposal are GFG [17, 18], GPSR [20].
One of the key challenges in geographic routing is how to deal with dead-ends, where greedy routing fails because a node has no neighbours closer to the destination; a variety of methods (such as perimeter routing in GPSR/GFG) have been proposed for this. More recently, GOAFR+ [37] proposes a method for routing around voids that is both asymptotically worst case optimal as well as average case efficient.
Each of the subsequent algorithms assumes that:
a) each network node is informed about its own and about its neighbours' positions and
b) the source of a message knows the position of the destination.

3.4.1 GFG - Greedy Face Greedy algorithm

In greedy routing algorithm (that has close performance to the shortest path algorithm, if successful), the sender node or an intermediate node currently holding the message *m*, forwards *m* to one of its neighbours that is the closest to the destination. The algorithm fails if the forwarding node does not have any neighbour that is closer to destination than itself.
FACE algorithm guarantees the delivery of *m* if the network, modelled by unit graph, is connected [17].
Unit graphs are a reasonable mathematical abstraction of wireless networks in which all nodes have equal broadcast ranges. Two nodes, in a network modelled as a unit graph, are neighbours if the Euclidean distance between their coordinates in the network is at

most R, where R is the transmission radius which is equal for all nodes.
GFG algorithm combines greedy and *FACE* algorithms.
Greedy algorithm (like GEDIR for example, which uses beacon-broadcast) is applied as long as possible, until delivery or a failure. In case of failure, the algorithm switches to *FACE* algorithm until a node closer to destination than last failure node is found, at which point greedy algorithm is applied again. Past traffic does not need to be memorized at nodes.
GFG algorithm, as defined, runs greedy algorithm until a node x is reached that has no closer neighbour than itself to destination *D*. Let *d* be distance from *C* to *D*. At *x*, the algorithm converts to the *FACE* algorithm, which has long path but guarantees delivery. When *FACE* algorithm reaches a node *A* whose distance to *D* is < *d*, algorithm switches back to greedy algorithm at *A*. This conversion may occur for an unlimited number of times, until message is delivered.

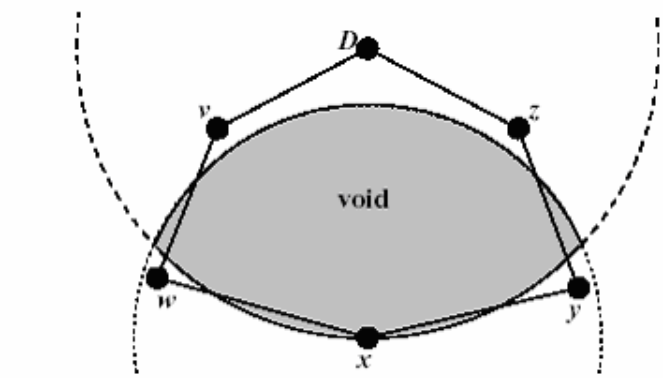


Figure 4.3*. Node x's void with respect to destination D.*

The above figure represents a topology in which the only route to a destination requires a packet move temporarily *farther* in geometric distance from the destination D.
Here, *x* is closer to *D* than its neighbours *w* and *y*. Although two paths, (*x*->*y*->z->*D*) and (*x*->*w*->*v*->*D*), exist to *D*, *x* will not choose to forward to *w* or *y* using greedy forwarding.
x is a local maximum in its proximity to *D*.

The Right-Hand Rule

The *intersection* of *x*'s circular radio range and the circle about *D* of radius |*xD*| (that is, of the length of line segment *xD*) is empty of neighbours. From node *x*'s perspective, the shaded region without nodes is a *void*. *x* seeks to forward a packet to destination *D*

beyond the edge of this void. Intuitively, *x* seeks to route *around* the void; if a path to *D* exists from *x*, it doesn't include nodes located within the void (or *x* would have forwarded to them greedily).
This is where the *right-hand rule* for traversing a graph comes into play.
This rule states that when arriving at node *x* from node *y*, the next edge traversed is the next one sequentially counter clockwise about *x* from edge (*x*,*y*).
The right-hand rule traverses the interior of a closed polygonal region (a *face*) in clockwise edge order. The rule traverses an exterior region in counter clockwise edge order.
Face algorithm seeks to exploit these cycle-traversing properties to route *around voids*.
In figure 4.3, traversing the cycle (*x*->*w*->*v*->*D*->*z*->*y*->*x*) by the right-hand rule amounts to navigating *around the pictured void*, specifically, to nodes closer to the destination than *x* (in this case, including the destination itself, *D*). The sequence of edges traversed by the right-hand rule is called a *perimeter*.
The *Relative Neighbourhood Graph (RNG)* and *Gabriel Graph (GG)* are two planar graphs; an algorithm for removing edges from the graph that are not part of the RNG or GG would yield a network with no crossing links.

3.4.2 GPSR - Greedy Perimeter Stateless Routing algorithm

From the first theoretical version of GFG [17], Karp and Kung [20] implemented an algorithm by including MAC layer considerations and experiments with moving nodes: this algorithm is GPSR. GPSR [20] uses sparser relative neighbourhood graph instead of its supergraph, Gabriel graph, used in [17] and changes faces before edge crossings instead of doing it afterwards.

The *greedy perimeter stateless routing (GPSR)* protocol [20] assumes that each mobile host knows all its neighbours' locations (with direct links).
It assumes bidirectional radio reachability. In [20] simulation has been conducted on a network that uses IEEE 802.11 links. 802.11 wireless network MAC [38] sends link-level acknowledgements for all unicast packets, so that all links in an 802.11 network must be bidirectional. A simple beaconing algorithm provides all nodes with their neighbours' positions: periodically, each node transmits a beacon to the broadcast MAC address, containing only its own identifier (*e.g.*, IP address) and position. Position is encoded as two four-byte floating point quantities, for *x* and *y* coordinate values.
Upon not receiving a beacon from a neighbour for longer than timeout interval *T*, a GPSR router assumes that the neighbour has failed or gone out-of-range, and deletes the

neighbour from its table.
This beaconing mechanism does represent pro-active routing protocol traffic, avoided by DSR [6] and AODV [3, 4]. To minimize the cost of beaconing, GPSR piggybacks the local sending node's position on *all* data packets it forwards, and runs all nodes' network interfaces in promiscuous mode, so that each station receives a copy of all packets for all stations within radio range. At a small cost in bytes (twelve bytes per packet), this scheme allows all packets to serve as beacons. When any node sends a data packet, it can then reset its inter-beacon timer. This optimization reduces beacon traffic in regions of the network actively forwarding data packets.
Like DSR [6] does, GPSR disables MAC address filtering to receive copies of all packets for all stations within its radio range. All packets carry their local sender's position, to reduce the rate at which beacon packets must be sent, and to keep positions in neighbour lists maximally current in regions under traffic load.
Authors of [20] say they could make GPSR's beacon mechanism fully reactive by having nodes solicit beacons with a broadcast "neighbour request" only when they have data traffic to forward.

The location of the destination host is also assumed to be known in advance.

Different from the LAR protocol and like GFG, the GPSR protocol does not need to discover a route prior to sending a packet. A host can forward a received packet directly based on local information.
Two forwarding methods are used in GPSR: *greedy forwarding* and *perimeter forwarding*.
When host the source host *S* needs to send a packet to a host *D*, it picks from its neighbours one host that is closest to the destination host and then forwards the packet to it.
The host receiving the packet follows the same greedy forwarding procedure to find the next hop. This is repeatedly used until host *D* or a local maximum host is reached.
A *local maximum host* is one that finds no other hosts that are closer to *D* than itself. A host *T* is a local maximum if all its neighbours are farther from *D* than itself. Therefore, the greedy forwarding method will not work here. When this happens, the perimeter forwarding method is used to forward the packet. The perimeter forwarding method works as follows.
The local maximum host first "planarizes" the graph representing the network topology. A graph is said to be *planar* if no two edges cross. The graph may be transformed into a *relative neighbourhood graph (RNG)* or a *Gabriel graph (GG)*.
Both RNG and GG are planar graphs. After the graph is planarized, the local maximum host T can forward the packet according to a *right-hand* rule to guide the packet along the perimeter of a plane counter clockwise. As the packet is forwarded to a host, say W,

we know that we are closer to *D* (as opposed to the location of host *T*). Then the greedy forwarding method can be applied again and the packet will reach destination *D*.
Overall, these two methods are used interchangeably until the destination is reached. The GPSR is a stateless routing protocol since it does not need to maintain any routing table.

3.4.3 About GFG - Greedy Face Greedy and GPSR

S. Datta, I.Stojmenovic and J.Wu [18] state that changes made in GPSR over the basic GFG make GPSR algorithm worse than previously published GFG algorithm [17].
Moreover GPSR does not take into account the improvements made to GFG in [18]: Datta, Stojmenovic and Wu [18] successively improved GFG by applying the concept of internal nodes to improve the delivery rates of *GFG* algorithm and proposed a shortcut procedure that allows each node to find out few next hops in *FACE* algorithm and forward the message directly to the last of these hops.
These changes aim to improve GFG by reducing the number of hop counts visited while algorithm is in FACE mode and reducing time GFG runs FACE algorithm by defining a *sooner-back* procedure to return back to greedy algorithm in few hops sooner than original GFG does.
Even Barriere, Fraigniaud and Narayanan [19] improved the nearly stateless routing protocol GFG by developing a model to cope with instability in the transmission ranges of nodes in order to achieve a better quality and robustness.
Author of GPSR [20] say that authors of [17] analyzed the increase in path length over shortest paths when traversing a graph using *only* perimeters, that they did not present a routing protocol, nor simulated a network at the packet level, and assumed that all nodes were stationary and reachable.

3.4.4 GOAFR+

Like GFG and GPSR, the GOAFR+ [37] algorithm is a combination of *greedy routing* and *face routing*. Whenever possible the algorithm tries to route greedily, that is by forwarding the message at each intermediate node to the neighbour located closest to the destination.
The message can however reach a “dead end", a node without any “better" neighbour. Such cases are overcome by the employment of face routing, which explores the boundaries of faces of the planarized network graph.
GOAFR+ uses an “early fallback" technique to return to greedy routing as soon as possible. This is the key aspect of this algorithm.
GOAFR+ applies a face routing technique which proceeds towards the destination by exploring the boundaries of the faces of a planarized network graph, employing the local right hand rule. Additionally the algorithm restricts itself to a searchable area occasionally being resized during algorithm execution. With this approach the algorithm becomes asymptotically optimal with respect to its execution cost compared with the cost of the optimal path.
Having escaped the local minimum, the algorithm continues in greedy mode. Since greedy forwarding is - above all in dense networks - more efficient than face routing in the average case, the algorithm should, for practical purposes, fall back to greedy mode as soon as possible. From studies conducted on algorithm which uses a Face algorithm, the authors observed that algorithm variants with heuristics employed for early fallback to greedy mode (such as the “First Closer" heuristic having the algorithm resume greedy routing as soon as meeting a node closer to the destination than where the current face routing phase started) lose their asymptotic optimality with respect to the shortest path. It appeared that, once in face routing mode, an algorithm is required to explore the complete boundary of the current face in order to be asymptotically optimal.
Contrarily to this conjecture, the *GOAFR+ algorithm does not necessarily explore the complete face boundary in face routing mode and yet does conserve asymptotic optimality.*
For this purpose the algorithm employs two counters p and q to keep track of how many of the nodes visited during the current face routing phase are located closer (p) and how many are not closer (q) to the destination than the starting point of the current face routing phase; as soon as a certain fallback condition holds, GOAFR+ directly falls back to greedy mode. Besides being asymptotically optimal, however, simulations show that in the average case GOAFR+ even outperforms the best (not asymptotically optimal) previously considered algorithms.

3.4.5 Intermediate Node Forwarding - INF

The *intermediate node forwarding* (INF) [42] mechanism is a probabilistic solution for routing around bad geographic topologies via intermediate geographic locations. While aforementioned solutions assume that nodes have identical radio propagation, INF works on a restricted set of situations but makes assumptions that better match reality.
It is introduces a mechanism that need some state keeping but can allow geographic routing to recover faster in case of failure.

Geographic forwarding works best when the spatial density of network nodes is high relative to the radio coverage.
Otherwise, cases where geographic forwarding's greedy choices fail to find routes are easy to find. Geographic forwarding will fail at a node when the packet has to travel backwards around a topology *hole*—when no neighbour is closer to the destination. The device currently forwarding the packet has no routes to any devices that are closer than itself to the packet's destination. A practical geographic forwarding system must handle these cases, as node distributions will vary unpredictably in the real world.
Although there are theoretically guaranteed techniques [17, 18, 20, 21] to route around topology holes, they assume that all nodes have radios with identical ranges. This is not likely to be even approximately true, since obstructions and interference drastically modify radio ranges. The intermediate node forwarding (INF) technique provides a probabilistic solution for handling topology holes, and does not assume uniform radio ranges.
[19] however address transmission problems with the GFG protocol.

The basic idea is that when using INF, nodes pick random intermediate points through which to forward their packets. Packets are routed from the source to the intermediate point using geographic forwarding and from the intermediate point to the destination using geographic forwarding again. The intermediate location serves as a weak source route. Eventually, an intermediate point can be chosen so that packets can be sent far enough out of the way of holes and other bad network topologies.
Nodes do not normally send packets using INF. However, if packets are unable to reach a destination using geographic forwarding, a sending node starts using INF for that destination: it picks an intermediate location and labels packets to the destination with the intermediate location. If packets still fail to reach a destination using INF, the node picks a new intermediate location. For the situations in which this approach works, the source node will eventually pick an intermediate point that causes packets to be routed around an intervening hole.

3.5 About Geographic Routing in Vehicular Ad Hoc Networks

At this point geographic routing seems to be the best choice in order to perform routing of packets in a wireless environment. And this type of routing seems to have only advantages and be applicable and effective "as is".
But that's not true for all environments.
And in particular for a vehicular environment better can be done, given its singular characteristics.

Environment topology greatly affects routing performance and often many scenarios' configurations can bring the aforementioned algorithms to not deliver packets in a reasonable time interval or/and fail.
This is particularly true for road scenarios where road topology greatly affects inter-vehicles network's topology. However a detailed knowledge of road topology in each vehicle can cope with this lack and can make inter-vehicle routing more efficient by overcoming this aspect.
Overcoming this restriction and instead using it to improve geographic routing can be done with a detailed knowledge of the underlying road topology, but this poses others requirements to mobile nodes.
A detailed knowledge of the environment's topology is prohibitive for other wireless scenario, but inter-vehicle networks have fewer constraints than other mobile networks (regarding space, power, moneys) and the availability of a detailed knowledge of the underlying road topology is feasible thanks to digital road maps.
Thus, in order to perform a correct and efficient geographic routing in wireless networks, many needs arise, above all the need for GPS devices (or other mechanisms of location tracking) and digital road maps. These are things that pose great constraints on the development of mobile ad hoc networks, but if there is an environment which can satisfy these needs this is the inter-vehicles communications scenario.

Analyzing position based routing (implemented by localized algorithms) we can see that the stateless nature of geographic forwarding is also its biggest constraint. While the stateless strategy helps geographic forwarding reducing routing overhead caused by topology updates, its lack of global topology knowledge prevents a mobile node from predicting topology holes as well as forwarding failures.
Although there are some methods proposed to route around the holes like GFG, GPSR and INF, they are used only *after* a geographic forwarding fails, incurring extra cost in detecting the failure and searching for new routes. Moreover, geographic forwarding

protocols often assume a uniform distribution of nodes, thus the topology holes only appear occasionally. Nevertheless, this assumption can often be violated in the real world.
Algorithms like GPSR achieve a higher scalability than topology-based routing protocols that rely on end-to-end state concerning the whole forwarding path, however, the advantage in scalability has its price: that algorithms will greedily forward a packet for potentially many hops, before a greedy forwarding failure is recognized or the packet is considered to be undeliverable.
Thus, the stateless strategy can only make locally optimal forwarding decisions rather than global optimal.
Moreover, the perimeter forwarding requires strictly identical radio ranges of nodes to construct a connected planar graph. This requirement is not always fulfilled in the reality due to obstructions and interferences.
Forwarding often takes physical distance as the basis for forwarding decisions: data packets are forwarded to the neighbour node with the shortest physical distance to the destination node as long as such a neighbour node can be found in the radio range of the current node.
However, *in the real world, positions have more meaning than just coordinates, taking their spatial environments into account.* Therefore, the correctness of forwarding decisions based *only* on physical distances is questionable in situations with holes in node distribution, since the topological assumption described above is likely to be violated.

Figure 4.4 shows a snapshot of an ad hoc network consisting of cars driving on the roads. It is obvious that the cars are not uniformly distributed in the whole plane.
The circle centred on the node S indicates its radio range. Node S wants to forward a packet to the destination node D, while two nodes A and B are currently located in its radio range.
As a basic prerequisite of geographic forwarding each node knows its current position, the position of its immediate neighbours and the approximate location of the packet's destination. We can assume all nodes have identical radio range and a connected path exists between source and destination.
According to the geographic forwarding strategy used in GPSR, GFG and INF, S will forward the data packet to A since it has the shortest Euclidian distance to the destination.

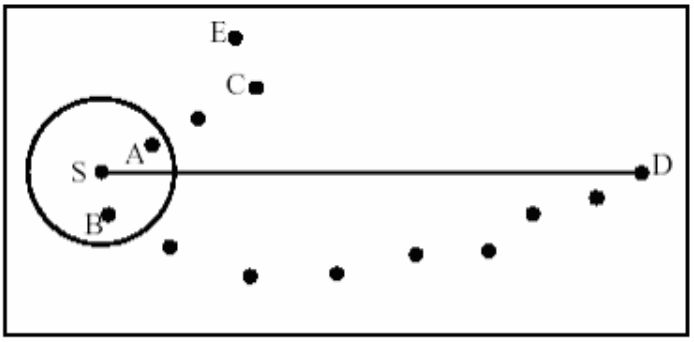


***Figure 4.4**. Geometric view of network.*

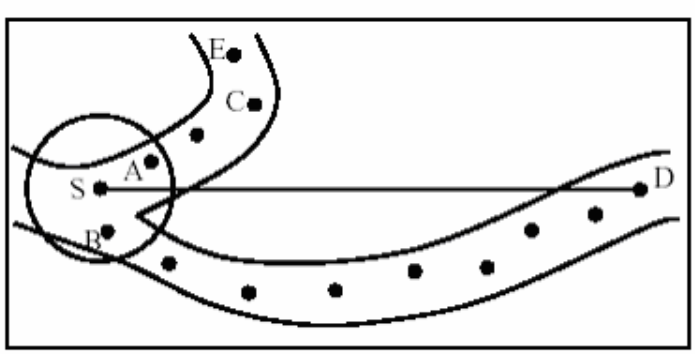


***Figure 4.5**. Topological view of network.*

However, as figure 4.5 reveals, this decision is far from being optimal, in fact it is wrong. Considering the underlying road structure shown in figure 4.5, we can understand the cause of the non-uniform nodes' distribution. As we see, all nodes only are located along roads; a big topology hole thus occurs at the fork of the road.
Although node A is physically closer to the destination than node B, it is on the branch that goes further away from the destination instead of approaching it. So actually node B is the right choice for packet forwarding at node S. However, node S is not aware of this fact, since the underlying spatial environment is not taken into account. Thus positions are still considered at the geometric level as shown in figure 4.4.
Using the greedy forwarding strategy, S forwards the packet to A, which will leads to a greedy failure at C, as the positions of nodes in figure 4.4 indicate. Perimeter forwarding or INF will then be started for recovery:

- In perimeter forwarding, C will forward the packet to E, trying to route round the topology hole.

- In INF, S will receive a NAK message from C and select a location between S and D randomly as intermediate destination. It is possible that the selected location is located above the line S-D.

Nevertheless, both perimeter forwarding and INF can fail if there is no connected link existing above the line S-D.

The aforementioned example illustrates the impact of spatial environments on both geographic forwarding and recovery methods: while geographic forwarding fails at constant topology holes due to spatial constraints, the proposed recovery methods may also fail even if a connected path from the source to the destination exists in the network.

Most performance studies of routing protocols so far assume topology holes to occur rarely. However, this assumption is only valid if the network has a high density *and* nodes are uniformly distributed in the whole area.

The described scenario shows that spatial constraints can cause frequent topology holes even with high network density. Although the one described was a road scenario, the impact of spatial constraints on routing can be found in many other scenarios, such as pedestrians on the street, ships in the river or people in the building, etc.

Spatial constraints and obstacles such as road infrastructures and buildings make the non-uniform distribution of nodes more likely to be the rule than exception.

The basic idea is to make use of the spatial knowledge to predict and avoid forwarding failures at constant topology holes caused by spatial constraints.

If a node can predict such topology holes, it can optimize its forwarding decision accordingly to avoid routing to fail. The topology holes caused by natural or man-made spatial constraints, e.g. lakes or road intersections, can be quite predictable with the external knowledge of spatial environments.

The utilization of spatial knowledge is generic and can be used to enhance any geographic forwarding approach, like Greedy Perimeter Stateless Routing (GPSR) or GFG.

Cars driving on the road can be simply modelled as nodes moving on the edges of a graph. Using digital maps, a graph can be constructed to model the major topology of the road. Moreover, external spatial knowledge can also help to speed up the recovery process in case of a forwarding failure.

The authors of [41] investigate the utilization of spatial knowledge, such as digital maps used in navigation systems. Their solution is to *proactively* avoid routing failures caused by topology holes.

Geographic routing is the best choice of routing because it does not maintain state, it is a localized mechanism, thus is adapt for large and highly mobile environments. It tries to reach a global optimum by selecting local optimums. However in a real world scenario a global optimum often cannot be reached by simply making consecutive local optimum choices. A global view of the underlying environment's topology must be used in order to perform a correct and also better geographic routing. This can be done in a vehicular network by exploiting digital detailed road maps.
Each node, which performs position based routing in order to deliver a packet, controls the road map, which in this scenario represents someway the network's topology and forwards the packet in a localized manner but taking into account a global view of the network, for which there is no need of inter-vehicles communications and routing state maintenance.
Digital maps are then a complement for geographic routing to work better and correctly.

Chapter 5

GeoCasting

1. GeoCasting

Geocasting is a variation on the notion of multicasting. The goal of a geocasting protocol is to deliver data packets to a group of nodes that are within a specified geographic area, i.e. the geocast region. A geographic area is associated with each geocast message and a geocast message is delivered to all the nodes within the specified geocast area. Thus, geocasting may be used for sending a message that is likely to be of interest to every one in a specific area. Unlike traditional multicasting schemes, geocasting implicitly defines a *group* as the set of hosts within the selected geocast region, *geocast group*. A host is a member of a geocast group if it is inside the geocast region (which is specified in each packet).
Thus, if a host resides within the geocast region at a given time, it automatically becomes a member of the corresponding geocast group at that time, and will receive the geocast packet.
In order to determine group membership, each node is required to know its own physical position, i.e. its own coordinates and these can be obtained using the Global Positioning System (GPS) [11, 12, 13, 14]. As written before, it is foreseeable that GPS devices will be deployed in almost every user terminal, above all in vehicles, which have very few constraints.

A destination geographic address would be represented by some closed polygon such as a:

1 circle(center point, radius)

2 polygon(point*1* , point*2* , ... point*n*)

where each vertex of the polygon is represented using geographic coordinates. This notation would be used to send a message to anyone within the specified geographic area defined by the closed polygon.

According to [12], possible application uses can be:

- Geographic messaging: sending a message selectively only to specific sub areas defined by latitude and longitude. For example an emergency message to everyone who is currently in a specific area, such as a building, train station or a highway.

- Geographic services and advertising: Providing a given service or advertising

only to clients who are within a certain geographic range from the server (which may be mobile itself); such everyone within a mile from the server.

- "Who is around" services: finding out who is currently present in a specific geographic area defined by an arbitrary polygon.

In contrast to multicast, which enables a packet to be sent to an arbitrary group of nodes, for example to all nodes that wish to subscribe to a news channel, a geocast group is only defined by a geographic region. Geocasting is a type of multicasting and can be implemented with a multicast service by simply defining the multicast group to be a certain geographic region, as described with the GeoNode approach [11, 12]. However, this leads in most cases to non-optimal protocols, especially in ad hoc networks, where geographic information can be used to make routing more efficient.

2. Geographic routing and Geocasting for inter-vehicles communications

A position based routing is a mechanism which automatically arises from applications' requirements in vehicular networks.
So called *x-cast* (broadcast, multicast…) routing methods will have great importance: security applications above all will be based on geographic position and for many other applications a fundamental issue will be reaching all the vehicles in a given geographic area.
Geographic position can be also utilized to perform or improve routing techniques even for other types of applications.
Thus applications' requirements drive the choice of a routing algorithm toward a position based routing algorithm (i.e. geographic routing) for inter-vehicles communications.
We have seen that a native position based routing gives the best results compared with other routing techniques, then we can successfully associate geographic routing to inter-vehicles communications; the only constraint would be the need for GPS devices, but this is of no matter in vehicular networks.
Thus, routing in a vehicular ad hoc network can exploit a native geographic routing mechanism, which also gives the best routing results.

While Geocasting is an important service, it is more likely that we will multicast rather than broadcast into the geographic areas. For example, reaching all *motorists* on a specific highway, or all *police cars*, rather than reaching *everybody* will be more useful. This aspect is called Geomulticasting and is another important feature an underlying geographic routing must support. It will be discussed later because it needs an underlying Geocast routing protocol to deliver messages to the destination area.

Now the focus is on Geocasting, so a classification of Geocasting protocols will be developed and a description of them will be given.

3. Geocast protocols

Geocast protocols can be classified in protocols for *infrastructured* networks such as the current Internet or for *ad hoc* networks and in protocols based on *flooding* or in protocols based on forwarding a geocast packet on a particular *routing path* .
In this section we focus on protocols for ad hoc networks.

All the present ad hoc geocasting protocols work under the following two assumptions:
1) Each node is supposed to know its own location
2) Whenever a node in the geocast region receives a geocast packet, it will flood the geocast packet to all of its neighbours.

One effect of these assumptions is that a geocast protocol only needs to work on having one node in the geocast region receiving the geocast packet from the source.
Since all the nodes in the geocast region share information among each other by flooding, the difference between flooding and non-flooding approaches is about how they transmit information from a source to one or more nodes in the geocast region.
If the source is within the geocast region, it will flood each geocast packet within the geocast region.

The challenging problem in geocasting is distributing the packets to all the nodes within the geocast region with high probability but with low overhead.

None of the proposed protocols are based on naive flooding, that is, flooding of a whole network without trying to limit the flooding area. However this is a likely solution and such a naive protocol is called the *simple flooding* approach.
Simple flooding was not proposed as a geocast routing protocol, but it is useful for comparison with other geocast protocols and it is a building block for many of them.
A *simple flooding* geocast algorithm works as follows. A node broadcasts a received packet to all of its neighbours, provided that this packet was not already received before, in order to avoid loops and endless flooding. A node delivers a packet if its own location is within the specified destination region, which is included in each geocast packet. This is a simple and robust but not efficient approach, since location information is not used for forwarding in order to reduce the number of packets.

As stated previously, the classification of geocasting protocols is based on whether a geocast protocol uses flooding or a variant of flooding to forward data from the source to the geocast region or produces routes to send data from the source to the geocast region.

Two categories of geocasting protocols can be given:

- Data-transmission oriented protocols [27] (also called *directed flooding protocols* [28]).
- Routing creation oriented protocols [27] (also called *non flooding protocols* [28]).

Directed flooding tries to limit the message overhead and network congestion of naive flooding by defining a forwarding zone, which comprises a subset of all network nodes. The forwarding zone includes at least the sender of a geocast message and the destination region of the message and additionally should include a path between the sender and destination region. If the last condition is not fulfilled, protocols either have to increase the forwarding zone or fall back on simple flooding.
An intermediate node forwards a packet only if it belongs to the forwarding zone.
Directed flooding protocols differ in the manner in which the forwarding zone is defined, however they make use of flooding or a variant of flooding to forward geocast packets from the source to the geocast region.
Known algorithms of this type are LBM [29, 30, 28], Voronoi diagram based algorithms [28], GEOGRID [31, 28].

Non-flooding approaches do not use flooding to reach the destination region of a geocast operation but other routing approaches. This behaviour refers only to the wide-area routing before the destination region of a geocast is reached. Inside the destination region, regional flooding may still be used even for protocols characterized as non-flooding. Routing-creation oriented protocols create routes from the source to the geocast region via control packets.
GEOTORA, GeoNode are *non flooding* protocols.
GeoNode [11, 12] requires an infrastructured network. It assumes a network with a cellular architecture, with a GeoNode, a geographic aware router, assigned to each cell.

Another class of Geocasting approaches is represented by the *Mesh-based protocol* [28, 52]. This protocol's approach has both the aspects of directed flooding and non flooding approaches. In fact Mesh uses directed flooding to discover redundant paths to the destination region, but instead of flooding geocast packets tries to create redundant routes via control packets. The actual geocast packet's payload is sent on the discovered paths, called mesh, without a network-wide flooding.
The initial step is only used to create the mesh rather than for sending the actual geocast payload. After a node inside the destination region received the initial packet to join the mesh, a unicast reply is sent back to the sender on the reverse path and flooding is stopped.

This requires that state information is maintained on each intermediate node or that the route is recorded in the flooded packet.

All of these techniques eventually reach one or more nodes in the geocast region.
The following figure represents a taxonomy for Geocasting algorithms.

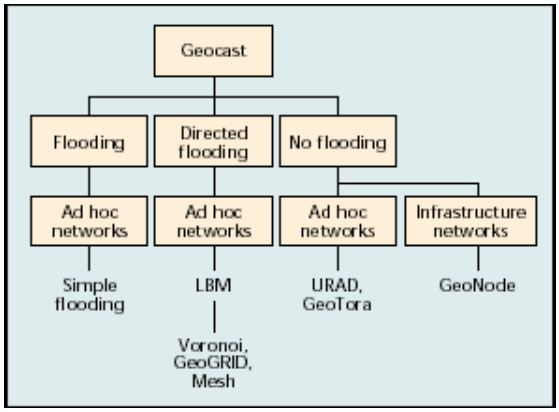


Figure 5.1*. Geocasting taxonomy*

Below will be described *directed flooding protocols* (*data-transmission oriented protocols*) and *routing creation oriented protocols* (*non flooding protocols*).

3.1 Directed Flooding protocols

3.1.1 LBM – Location Based Multicast

LBM is the first published data-transmission oriented protocol [29, 30] and it is derived from the unicast protocol LAR [52].
LBM is essentially identical to flooding data packets, but it avoids flooding the whole network by defining a forwarding zone. Outside the forwarding zone a packet is discarded. A node determines whether to forward a geocast packet in one of two schemes [30].

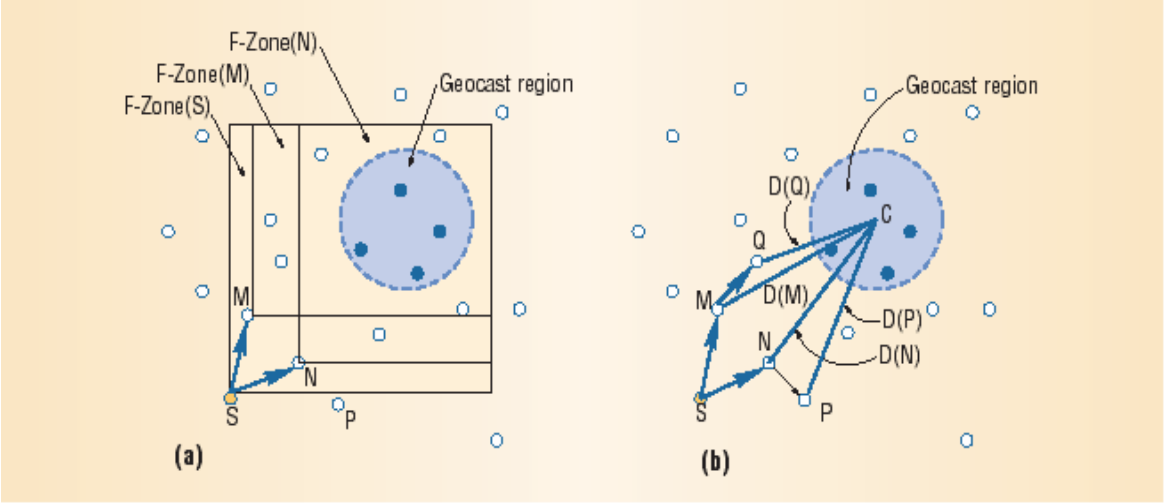


Figure 5.2 *Location based Multicast forwarding schemes.*
(a) The first LBM scheme defines a rectangular forwarding zone, one corner of which lies at the source node and spans the geocast region.
(b) The second LBM scheme uses a distance-based heuristic in which source node S defines the centre point C of the geocast region in the geocast packets. Each intermediate node decides whether to forward a geocast packet by comparing its distance to the packet sender's distance.

The first scheme defines a forwarding zone that includes at least the destination region and a path between the sender and the destination region. An intermediate node forwards a packet only if it belongs to the forwarding zone.

By increasing the forwarding zone, the probability of reception of a geocast packet at all destination nodes can be increased; however, overhead is also increased.
Thus, how to define the forwarding zone becomes the key point of this scheme.
The forwarding zone can be the smallest rectangular shape that includes the sender and the destination region, possibly increased by a parameter d to increase the probability for message reception. A description of the geocast region is included in each geocast packet. A host, say Z, on receiving the packet, compares the geocast region's coordinates with its own location. If host Z is within the geocast region it will accept the packet. Also, Z will propagate the packet to its neighbours, if it has not received the packet previously (repeated reception of a packet is detected using sequence numbers). If host Z is located outside the geocast region and the packet was not received previously, it just broadcasts the packet to its neighbours.

The second scheme does not define a forwarding zone explicitly, instead weather a geocast packet should be forwarded is based on the position of the sender node at the transmission of the packet and the position of the geocast region: this scheme defines the forwarding zone by the coordinates of the sender, the destination region, and the distance of a node to the centre of the destination region. A node receiving a geocast packet determines whether it belongs to the forwarding zone by calculating its own geographic distance to the centre of the destination region. If its distance decreased by d is not larger than the distance stored in the geocast packet, which is initially the sender distance, the geocast packet is forwarded to all neighbours and the packet sender's distance is replaced by the calculated own distance.
In other words, a node forwards a packet if it is not farther away from the destination region than the one-hop predecessor of the packet increased by d. Finally, a geocast packet is forwarded to all neighbours if the one-hop predecessor is located inside the destination region.

3.1.2 Voronoi diagram based algorithms

Voronoi-diagrams-based routing [28, 52] improves the LBM [29, 30] approach, which fails if the forwarding zone is empty or partitioned. A new definition of the forwarding zone is given which overcomes these problems.
A neighbour of a sender belongs to the forwarding zone if and only if it is closest in the direction of the destination. As the destination is not defined by a single position but by an area, all possible positions of destinations inside the geocast region are considered. This leads to having several neighbours belonging to the forwarding zone. With this

definition of a dynamic forwarding zone, which in contrast to LBM takes the current neighbours position into account, an empty forwarding zone is avoided.
The neighbours belonging to the forwarding zone can be determined using the concept of Voronoi diagrams. A Voronoi diagram partitions the network in n Voronoi regions, where n is the number of neighbours. Each neighbour is associated with one Voronoi region. The Voronoi region of a neighbour consists of all nodes that are closer to this neighbour than to any other neighbour.
If a node holds a geocast packet, it starts with determining the Voronoi diagram. The Voronoi partitions intersecting with the geocast destination region belong to the forwarding zone and are selected for geocast forwarding. Inside the destination region, flooding can be used. In fact, any protocol can be used that can be independent of the protocol used outside the destination region.
The major advantage of Voronoi-diagrams based routing is that empty forwarding zones are avoided.
However, flooding overhead is still high and additional computation overhead is introduced by determining the Voronoi partitions.

3.1.3 GEOGRID

GeoGRID [31] is a geocast protocol modified from the unicast GRID protocol.
The GRID [52] protocol divides the network area into several nonoverlapping squares called grids.
GeoGRID uses location information, which defines the forwarding zone, and elects a special host (i.e., *gateway*) in each grid area responsible for forwarding the geocast packets.
Geocasting messages are sent in a grid-by-grid manner through grid leaders. However, in GeoGRID, no spanning tree or routing path needs to be established before geocasting. Instead, a connectionless mode is adopted.
Because only gateways in every grid within the forwarding zone will rebroadcast the received geocast packets, gateway election becomes a key point of this protocol.
Two approaches are suggested to propagate geocast packets.
The first approach is *flooding-based*. Every grid leader in the forwarding zone will forward the geocast packets.
The second approach is *ticket-based*. Only the grid leader that holds a ticket will forward the geocast packets. The purpose of issuing tickets is to avoid blind flooding. The source needs to decide how many tickets will be issued. On their way to the geocast region, tickets may be split to different grids: if a gateway is not within the destination region, it will select up to three neighbouring gateways whose grids are closer to the

destination region and within the forwarding region. The geocast packet is then forwarded to the selected gateways and the tickets are evenly shared among them.
The idea is that each ticket is responsible for carrying one copy of the geocast packet to the destination region. Thus, by selecting a certain number of tickets the initial sender not only determines the overhead of geocast delivery but also the success probability of delivery.
The GeoGRID protocol can reduce network traffic, compared to LBM and Voronoi based algorithms and can achieve a high data arrival rate.

3.2 Routing creation oriented protocols

3.2.1 GEOTORA

The goal of GeoTORA [28] is to reduce the overhead of transmitting geocast packets via flooding techniques, while maintaining high accuracy. The unicast routing protocol TORA (Temporally-Ordered Routing Algorithm [V.Park, S.Corson, 2001]) is used by GeoTORA to transmit geocast packets to a geocast region. TORA is a distributed routing protocol based on a "link reversal" algorithm. It attempts to provide multiple routes to a destination, establish routes quickly and minimize communication overhead.
In GeoTORA, a source node essentially performs an *anycast* to any geocast group member (i.e., any node in the geocast region) via TORA. When a node in the geocast region receives the geocast packet, it *floods* the packet such that the flooding is limited to the geocast region.
Accuracy of GeoTORA is high, but not as high as pure flooding or LBM [29, 30]. One possible reason for the reduced accuracy is because only one node in the geocast region receives the geocast packet from TORA.
The overhead is relatively small, compared to data-transmission oriented approaches.

From the above solutions we can derive another solution, which is really natural and is the simpler one. It is also a valid solution, but in order to say it is the best, simulations and studies should be done.
In global flooding, the sender broadcasts a packet to its neighbours, and each neighbour, that has not received the packet before, broadcasts it to its neighbours, and so on, until the packet is received by all reachable nodes including the geocast region nodes. It is

simple but has a very high overhead and is not scalable to large networks. Ko and Vaidya [29, 30] proposed two geocasting algorithms to reduce the overhead, compared to global flooding, by restricting the forwarding zone for geocast packets. Nodes within the forwarding zone forward the geocast packet by broadcasting it to their neighbours and nodes outside the forwarding zone discard it. Each node has a localization mechanism to detect its location and to decide, when it receives a packet, whether it is in the forwarding zone or not.
To reduce the overhead further, GeoTORA [6] uses the unicast routing protocol TORA to deliver the packet to the region and then floods within the region.

We could use *any* unicast routing protocol for a MANET to perform geocast in this manner.
Since other unicast protocols perform better than TORA [3, 4, 5, 6, 15, 16], it is expected that a different Geo*PROTOCOL* would perform better than GeoTORA.

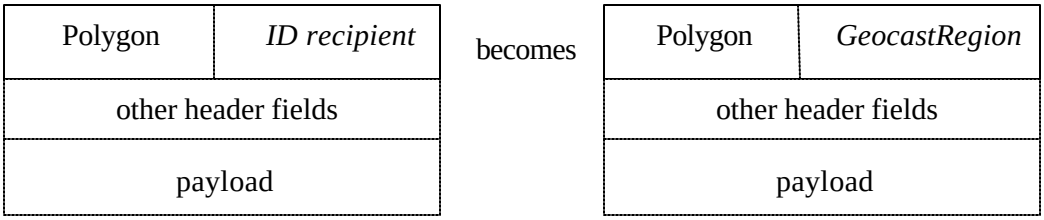
3.2.2 Geocasting by unicasting

Geographic routing, as previously seen, has several advantages: the state kept is minimum, nodes require only information from their direct neighbours so discovery floods and state propagation are not required, and accordingly it has low overhead and fast response to dynamics. Furthermore, since in geocasting, with the use of GPS or other location tracking mechanisms, nodes are aware of their locations, there are no extra costs for using geographic routing.
It should be observed that all protocols use the following assumptions assumptions:

1) each node is supposed to know its own location,
2) whenever a node in the geocast region receives a geocast packet, it will flood the geocast packet to all of its neighbours.

This means that in order to perform Geocasting, the protocols need to reach the geocast area, starting a trip from the sender. Once the destination has been reached, they use flooding to deliver the message and perform the Geocast operation.
The previously assumptions give geocasting an *anycasting* aspect: it suffices that one node inside the destination receives the message for Geocasting to work.
So we can reach the destination area with a geographic unicast routing like GPSR to reduce overhead and packets forwarding, performing a *GeoTORA-like* approach, but with the use of a more efficient underlying geographic unicast algorithm [18, 32, 33]. This is an approach that authors of [28] call URAD (Unicast Routing with Area

Delivery).
They suggest to fix as the destination the geocast region's centre (target area) and then send the message; inside the geocast region flooding will be performed to deliver the message.
Only the destination geographic area specifies the receivers: a polygon inside the geocasting area would be specified in each geocast packet to identify the receivers.
Each node inside the target area would then broadcast the packet to nodes outside the target area but inside the geocast region specified in the packet.
If a unicast packet has receiver position's coordinates and the identifier of the receiver, a geocast packet has the geocast region specified instead of receiver's identifier.



This way, geocasting becomes simpler than unicasting because a location service is of no need, there are no problems of destination movements and of updating recipient's position while a packet is travelling toward the destination: the recipients are specified by an area (...this won't move) fixed by the source and no specific node is the destination, but everyone who is in the specified area.

It is clear that a problem with this approach is one already faced by GeoTORA: only one node in the geocast region will receive a geocast packet and this node is then responsible for flooding the packet through the geocast region.
Because this method does not rely on flooding (which gives the best achievable robustness) it is less robust than LBM, Voronoi and Mesh based approaches.

An algorithm like GFG could be used to perform Geocasting in this manner [18] and an example of this kind of approach is GFG/GFPG [32] whose name is the same as the previously mentioned unicast GFG algorithm but is used by authors of [32] to refer to the Geographic-Forwarding-Geocast algorithm, an algorithm that is like GFG, but directly accounts the Geocast problem.
The observation made by the authors of [32] is the same expressed above: using unicasting to deliver packets to the region and using a geographic routing unicast algorithm instead of others routing protocols. The use of geographic routing is driven by

its several advantages over other routing approaches: the state kept is minimum, nodes require only information from their direct neighbours so discovery floods and state propagation are not required, and accordingly it has lower overhead and faster response to dynamics.
Three algorithms are presented in [32]: GFG, GFPG and GFPG*.

All the aforementioned geographic unicast algorithms used to perform Geocasting can be improved by exploiting detailed digital road maps, as explained in chapter 4, section 3.5.

3.2.3 Geographic-Forwarding-Geocast (GFG)

This algorithm is the GPSR algorithm modified so that a node wishing to send a geocast packet creates a packet and puts the coordinates of the region in the packet header. Forwarding outside the region is performed in the classical *greedy-perimeter (face)* mode employed in GPSR and GFG (Greedy-Face-Greedy) algorithms. The first node inside the region to receive the geocast packet starts flooding the region by broadcasting to all neighbours. Thus, this algorithm is an adaptation of GPSR where the flooding algorithm when a packet reaches the destination region is added: it is no more than that was exposed in the preceding section.
Simulation studies made in [32] show that in dense networks *without obstacles or gaps*, GFG is sufficient to deliver the packet to all nodes in the region. In addition, since in dense networks geographic routes are close to optimal routes (shortest path), GFG has almost the minimum overhead a geocast algorithm can have, which mainly consists of the lowest number of hops to reach the region plus the number of nodes inside the region itself.
In order for GFG to provide perfect delivery (i.e. all nodes in the region receive the geocast packet), the nodes in the region need to be connected together such that each node can reach all other nodes without going out of the region. In dense networks or due to obstacles, regions may have gaps such that a path between two nodes inside the region may have to go through other nodes outside the region.
In case of region gaps, GFG will fail to provide perfect delivery.
The GFPG algorithm was developed to overcome this limitation. Therefore the general idea of GeoCasting by unicasting remains valid here and all is done to forward the packet to the destination region is a packet forwarding with a geographic unicast algorithm. The receiver of the packet is a polygonal region and no location service is needed. It is the previously mentioned *anycast-like* part.
Inside the geocast region flooding is used to deliver the message.

The next discussed algorithm GFPG deals with the delivery of the message *inside* the geocast region and there is the contribution of [32] to geocasting.
Thus geocasting by unicasting can be a valid choice and the following is a solution that accounts the reliability of message delivery inside the geocast region once a packet has arrived inside it.

3.2.4 Geographic Forwarding Perimeter Geocast (GFPG)

GFPG is an algorithm that from simulation studies in [32] guarantees the delivery of a geocast packet to all nodes inside the geocast region, given that the network as a whole is connected. The algorithm solves the region gap problem in sparse networks, but causes unnecessary overhead in dense networks. Therefore, authors of GFPG present another practical version of the algorithm, called GFPG*, that provides good delivery at all densities and keeps the overhead low in dense networks. GFPG* is not guaranteed as the original version GFPG, but it seems to achieve efficient delivery like GFPG.

GFPG uses a mix of geocast and perimeter routing to guarantee the delivery of a geocast packet to all nodes in the destination region.
If there is a gap between two clusters of nodes inside the region, the nodes around the gap are part of the same planar face. Thus if a packet is sent in perimeter mode by a node on the gap border, it will go around the gap and traverse the nodes on the other side of the gap.
The idea of this algorithm is to use perimeter routing on the faces intersecting the region border in addition to flooding inside the region to reach all nodes.

Initially, similar to Geographic-Forwarding-Geocast (so like GPSR and Greedy-Face-Greedy), nodes outside of the geocast region use geographic forwarding to forward the packet toward the region. As the packet enters the region, nodes flood it inside the region. All nodes in the region broadcast the packet to their neighbours similar to Geographic-Forwarding-Geocast; in addition, nodes on the border of the region send perimeter mode packets to their neighbours that are outside of the region.
A node is a region border node if it has neighbours outside of the region. By sending perimeter packets to neighbours outside the region (notice that perimeter mode packets are sent only to neighbours in the planar graph not to all physical neighbours), the faces intersecting the region border are traversed. The node outside the region, receiving the perimeter mode packet, forwards the packet using the right-hand rule to its neighbour in the planar graph and that neighbour forwards it to its neighbour and so on. The packet

goes around the face until it enters the region again. The first node inside the region to receive the perimeter packet floods it inside the region or ignores it if that packet was already received and flooded before. This way if the region consists of separated clusters of nodes, a geocast packet will start at one cluster, perimeter routes will connect these clusters together through nodes outside the region, and each cluster will be flooded as the geocast packet enters it for the first time. This guarantees that all nodes in the region receive the packet, since perimeter packets going out of the region will have to enter the region again from the opposite side of the face and accordingly all faces intersecting the region will be covered.

3.2.5 GFPG*

Due to the perimeter traversals of faces intersecting the region, GFPG will cause additional overhead that may not be required especially in dense networks, where flooding or variant of flooding suffices to deliver the packet to all nodes inside the geocast region (GFG). Ideally perimeter routes should be used only when there are gaps inside the region in order to achieve a perfect delivery also in sparse networks but maintain a minimum overhead in dense networks.

GFPG* is an adaptation for GFPG in which perimeter packets are sent only when there is a suspicion that a gap exists: it was created to reduce overhead of GFPG in case of high nodes density, but still achieve a delivery quality like GFPG.
In this algorithm each node inside the geocast region divides its radio range into four portions as shown in figure 5.3(a) and determines the neighbours in each portion. This can be done easily, since each node knows its own location and its neighbours' locations. If a node has at least one neighbour in each portion, it will assume that there is no gap around it, since its neighbours are covering the space beyond its range and so it will not send a perimeter packet and will send only the region flood by broadcasting to its neighbours.

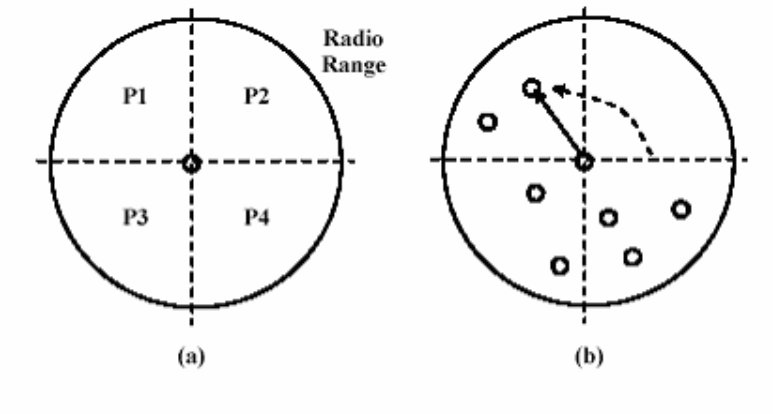


***Figure 5.3**. a) A node divides its radio range into four portions. b) If a node has no neighbours in a portion, it sends a perimeter packet using the right-hand rule to the first node counter clockwise from the empty portion.*

If a node has no neighbours in a portion, then it sends a perimeter mode packet using the right-hand rule to the first neighbour counter clockwise from the empty portion as shown in figure 5.3(b). Thus the face around the suspected void will be traversed and the nodes on the other side of the void will receive the packet. In this algorithm there is no specific role for region border nodes and perimeter packets can be sent by any node in the region, since the gap can exist and need to be detected anywhere. Therefore there are two types of packets in the region, flood packets and perimeter packets. Nodes have to forward perimeter packets even if that packet was flooded before. If a node receives a perimeter packet from the same neighbour for the second time, the packet is discarded, since this means that the corresponding face is already traversed. A node may receive the perimeter packet from different neighbours and thus forwards it on different faces.

Chapter 6

Multicast routing

Multicasting

Multicast consists in sending a packet from one sender to multiple receivers in a single operation.
Distributing significant amounts of identical data from a single sender to multiple clients can take considerable time and bandwidth if the sender must send a separate copy to each client.
For this reason the use of multiple point-to-point unicast connections is not the right way to implement multicasting: an explicit multicast support at the network layer is required.
Multicasting over a network allows a sender to distribute data to all interested parties while minimizing the use of network resources.
The difference between multicasting and separately unicasting data to several destinations is best captured by the *host group* model [73]: "a host group is a set of network entities sharing a common identifying multicast address, all receiving any data packets addressed to this multicast address by senders (sources)." This definition implies that, from the sender's point of view, this model reduces the multicast service interface to a unicast one.
Thus, the multicast model was proposed to reduce the many unicast connections into a multicast tree for a group of receivers.
The multicast definition also allows the behaviour of the group to be unrestricted in multiple dimensions: groups may have local (LAN) or global (WAN) membership, be transient or persistent in time, and have constant or varying membership.
With an explicit multicast support, a single packet is transmitted from the sending host and replicated at a network node (a router in Internet) whenever it must be forwarded on multiple nodes (in Internet, ongoing links) in order to reach the receiver.
The value of multicast features with routing protocols is even more relevant in ad hoc networks, because of limited bandwidth in radio channels.

1. Multicast Routing in Internet: An Overview

In order to perform multicasting in Internet, state information for a multicast connection must be created and maintained in routers that handle multicast packets sent among hosts in a so called multicast group: Internet multicast is not a connectionless service unlike the unicast case.
This, in turn requires a combination of signalling and routing protocols in order to set up, maintain and tear down connection state in the routers.
A single datagram transmitted by the sender is duplicated by routers within the networks. Only a single copy will ever traverse a link. On the other hand, considerable network layer support is needed to implement a multicast-aware network layer.
How to identify the receivers of a multicast datagram and how to address a datagram sent to these receivers are two problems to face with a multicast communication.
In the case of unicast communication, the IP address of the receiver is carried in each IP unicast datagram and identifies the single recipient, but in the case of multicast there are multiple receivers.
Including in a datagram the IP addresses of all receivers is not a feasible approach due to the amount of data to store in a datagram and because explicit identification of the receivers by the sender also requires that the sender knows the identities and addresses of all of the receivers.
For these reasons, in the Internet architecture a multicast datagram is addressed using *address indirection*. That is, a single identifier is used for the group of receivers and a copy of the datagram that is addressed to the group using this single identifier is delivered to all of the multicast receivers associated with that group.
The group of receivers associated with a multicast address is referred to as a *multicast group*.
Each host also has a unique IP unicast address that is completely independent of the multicast group in which it is participating.

There are two key aspects of multicast mechanisms in Internet: the first is a protocol to register hosts in a multicast group, it is IGMP [RFC 2236] and the second is a mechanism to coordinate the multicast routers throughout the Internet, this is accomplished by a network layer multicast routing algorithm such as PIM, DVMRP and MOSPF.

IGMP operates between a host and its directly attached router. It provides the means for a host to inform its attached router that an application running on the host wants to join

a specific multicast group. IGMP messages are carried in an IP datagram with an IP *protocol number* of 2.
Any host can join a multicast group at the network layer. A host simply issues a "membership_report" IGMP message to its attached router. That router, working in concert with other Internet routers will begin delivering multicast datagrams to the host. Joining a multicast group is thus receiver-driven. A sender need not be concerned with explicitly adding receivers to the multicast group but neither can it control who joins the group and therefore who receives datagrams sent to that group. Similarly there is no control over who sends to the multicast group. Network layer does not provide for filtering, ordering or privacy of multicast datagrams. This functionality should be provided by the upper layers.
In many ways, the current Internet multicast service model reflects the same philosophy as the Internet unicast service model: an extremely simple network layer with additional functionality being provided in the upper-layer protocols in the hosts at the edges of the networks.

The goal of a multicast routing algorithm is to find a tree that has attached hosts belonging to the multicast group. Multicast packets will then be routed along this tree from the sender to all of the hosts belonging to the multicast tree. The tree may contain routers that do not have attached hosts belonging to the multicast group.
In practice, two approaches have been adopted for determining the multicast routing tree. The two approaches differ according to whether a single tree is used to distribute the traffic for all senders in the group, or whether a source-specific routing tree is constructed for each individual sender:

- *Group-shared tree*. In the group-shared tree approach only a single routing tree is constructed for the entire multicast group.
- *Source-based tree*. In a source-based approach an individual routing tree is constructed for each sender in the multicast group. In a multicast group with N hosts, N different routing trees will be constructed for that single multicast group. Packets will be routed to multicast group members in a source-specific manner.

  These types of trees have been shown to be the most scalable way of supporting reliable multicast transmissions [70].

DVMRP, MOSPF [48], CBT and PIM are Internet multicast routing protocols [60].

DVMRP, Distance Vector Multicast Routing Algorithm, is the first routing protocol used in the Internet and the most widely supported multicast routing algorithm. It implements source-based tree with reverse path forwarding, pruning (in case a router receives multicast packets for a group to which no underlying hosts are subscribed) and

grafting (to “unprune” previously pruned multicast gr oups’ packets). DVRMP uses a distance vector algorithm that allows each router to compute the outgoing link (next hop) that is on its shortest path back to each possible source. This information is then used in the RPF algorithm [60].

MOSPF, Multicast Open Shortest Path First protocol [48], operates in an autonomous system (AS) that uses OSPF unicast protocol [60] for unicast routing. MOSPF extends OSPF by having routers add their multicast group membership to the link state advertisement that is broadcaste d by routers as part of the OSPF protocol. With this extension, all routers have not only complete topology information, but also know which edge routers have attached hosts belonging to various multicast groups. With this information, the routers within the AS can build source-specific, pre-pruned, shortest-path trees for each multicast group.

2. Multicast Routing in Ad Hoc Wireless Networks: An Overview

The multicasting communication model can facilitate effective and collaborative communication among groups. *Flooding* and *tree-based routing* represent two ends of the multicasting spectrum.

Flooding is a simple approach that offers the lowest control overheads at the expense of generating very high data traffic in the wireless environment.

The tree-based approach, on the other hand, generates minimal data traffic in the network, but tree maintenance and updates require many control traffic exchanges. Both flooding and tree-based approaches scale poorly.

The drawbacks of multicast trees in a mobile wireless networks are several: intermittent connectivity, traffic concentration, frequent tree reconfiguration, non shortest path in a shared tree, etc...

Multicast routing protocols for MANETs vary in terms of route topology, state maintenance, reliance on unicast routing, and other attributes [25, 26, 61, 69, 70, 71, 72, 73].

Most proposed multicasting protocols primarily exploit one or more specific characteristics of the MANET environment. These characteristics include variable topology, soft-state and state aggregations, knowledge of location, and communication pattern randomness. For example, mesh-based protocols exploit variable topology, stateless multicasting exploits soft-state maintenance, location-aided multicasting exploits knowledge of location, and gossip-based multicasting exploits randomness in communication and mobility.

2.1 Tree-based protocols

This kind of approach does not scale well in MANET environments with random medium/high nodes' relative mobility. However several tree-based algorithms for wireless networks exist: RBM, LAM, AMRoute, AMRIS, multicast AODV are tree-based algorithms [25, 52].

The Reservation Based Multicast (RBM) routing protocol builds a core based tree for each multicast group. It is a combination of multicast, resource reservation and admission control protocol where users specify requirements and constraints.

The Lightweight Adaptive Multicast (LAM) algorithm is a group shared tree protocol that does not require timed based messaging. Similar to other core based protocols, it suffers from disadvantages of traffic concentration and vulnerability of the core.

The Ad hoc Multicast Routing protocol (AMRoute) is a shared tree protocol which allows dynamic core migration based on group membership and network configuration.

The Ad hoc Multicast Routing protocol utilizing Increasing id-numberS (AMRIS) builds a shared tree to deliver multicast data. Each node in the multicast session is assigned an ID number and it adapts to connectivity changes by utilizing the ID numbers.

A multicast extension of the Ad Hoc On Demand Distance Vector (AODV) routing protocol has also been proposed. Its uniqueness stems from the use of a destination sequence number for each multicast entry. The sequence number is generated by the multicast group head to prevent loops and to discard stale routes.

2.2 Mesh-based protocols

The addition of redundant paths between on-tree nodes converts a multicast tree into a mesh topology.
The availability of alternative paths lets nodes deliver multicast packets regardless of link breakages.
Mesh-based protocols thus achieve higher robustness against node mobility.
Two mesh-based multicast routing protocols for MANETs are CAMP (1999) [52] and ODMRP (2002) [43, 52].

2.2.1 Core-Assisted Mesh Protocol - CAMP

CAMP [52] uses a shared mesh structure to support multicast routing in dynamic ad hoc networks. This structure ensures that the mesh includes the *reverse shortest paths*, the shortest paths from all receivers to the source.

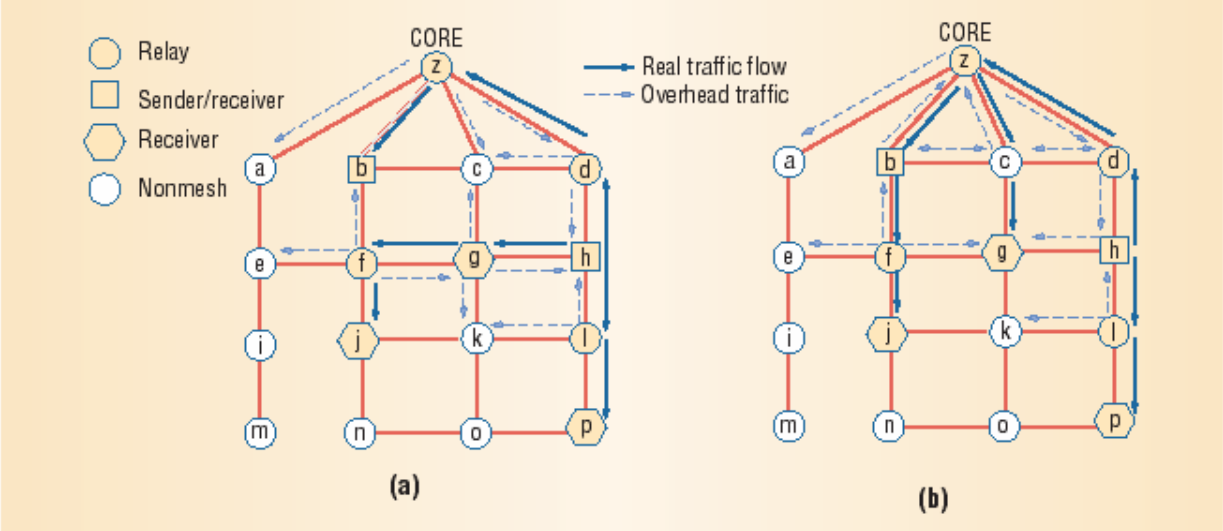


Figure 6.1*. CAMP multicast routing.*
(a) Data-packet forwarding from node ***h*** *using the core-assisted mesh protocol.*
(b) An equivalent shared tree, which uses a receiver-initiated router method.
The solid arrows indicate the flow of actual traffic and the dashed arrows indicate the broadcast traffic due to the broadcast nature of wireless links.

Figure 6.1 shows how the protocol forwards data packets from node *h* to the rest of the group. To prevent packet replication or looping in the mesh, each node maintains a cache to keep track of recently forwarded packets. Periodically, a receiver node reviews its packet cache to determine whether it is receiving data packets from those neighbours not on the reverse shortest path to the source. When such situations arise, the node sends a heartbeat message to its successor in its reverse shortest path to the source. When the successor is not a mesh member, the heartbeat message triggers a *push join* message, which includes all nodes along any reverse shortest path in the mesh.

CAMP uses cores to limit the control traffic needed to create multicast meshes. Unlike the core-based tree protocol, CAMP does not require that all traffic flow through the core nodes. CAMP uses a receiver-initiated method for routers to join a multicast group. If a node wishing to join such a group finds it has neighbours that belong to the group, it simply updates its multicast routing table and uses a standard update procedure to announce its membership.

When none of its neighbours are mesh members, the node either sends a join request toward a core or attempts to reach a group member using an expanding-ring search process. Any mesh member can respond to a join request with a join ACK, which propagates back to the request originator.

2.2.2 On-Demand Multicast Routing Protocol - ODMRP

ODMRP [43, 52] is a mesh-based multicast protocol: by building a mesh and supplying multiple routes, multicast packets can be delivered to destinations in a more robust manner than tree based approaches in the face of node movements and topology changes.

In order to establish a mesh for each multicast group, ODMRP uses the concept of forwarding group. The forwarding group is a set of nodes responsible for forwarding multicast data on shortest paths between any members' pair. A soft state approach is taken to maintain multicast group members and no explicit control message is required to leave the group. ODMRP applies on-demand routing.

Group membership and multicast routes are established and updated by the source "on-demand". Similar to on-demand unicast routing protocols, a request phase and a reply phase comprise the protocol. When a multicast source has packets to send, it floods a member advertising packet with data payload piggybacked. This packet, called JOIN QUERY is periodically broadcasted to the entire network to refresh the membership information and update the routes.

By flooding a member advertising packet, a source node starts building a forwarding mesh for the multicast group, collecting membership information at the same time.
When a node receives a no duplicate message requesting admission to the multicast group, it stores the upstream node identity and rebroadcasts the packet. When this request message packet reaches a multicast receiver, the receiver creates or updates the source entry in the *member table*. The system then uses the member table to prepare periodic control packets and broadcasts them via the receiver node.
The nodes relay the packets back toward the source along the reverse path that the member-advertising packet traverses. This process constructs or updates the routes from sources to receivers and builds a mesh of nodes, called the *forwarding group*.

The following is a table containing some characteristics of various Ad Hoc mobile multicast routing protocols.

Parameters	DVMRP	AODV	CAMP	ODMRP
Multicast delivery structure	Source-based tree	Core-based tree	Multicast mesh	Group-based
Use of centralized node	No	Yes (Multicast group leader	Yes (Core nodes)	No
Core node recovery	N/A	Yes	Yes	N/A
Routing scheme	Table-driven	On-demand	Table-driven	On-demand
Dependence on unicast routing protocol	No	No	Yes	No
Routing approach	Flat	Flat	Flat	Flat
Routing metric	Shortest path	Shortest path to another multicast member along the existing shared tree	Shortest path	Shortest path

Figure 6.2*. Characteristics of various Ad Hoc mobile multicast routing protocols.*

3. Location Aided Multicast Routing

In networks that can access the Global Positioning System (GPS), the network provides each node with location and mobility information.
Multicast protocols can use this information to improve protocol robustness and performance.
With GPS support, ODMRP can adapt to node movements and can use location and mobility information to estimate route expiration time, while receivers select the path that will remain valid longest. Sources can reconstruct routes in anticipation of route breaks, thereby making the protocol more resilient to node mobility.
GPS devices allow a sort of *mobility prediction* that can improve ODMRP algorithm.

In contrast to existing classical approaches, thanks to "location aware nodes" it is possible to implement an approach which does neither require the maintenance of state about a distribution structure nor does it resort to flooding of the data packets.
Each node that forwards a multicast packet can autonomously determine the neighbours that it should forward the packet to. This decision is based on information about the position of the destination nodes, the position of the forwarding node, and the position of the forwarding node's neighbours. It can be regarded as an adaptation of position-based unicast routing schemes such as GFG and GPSR to multicast routing.
But this approach is not effective or feasible neither for general environments nor for wide scenarios because each node must know the positions of all of the destinations, this information must be locally available or included in each packet's header.

In order to extend position-based routing to multicast, two key problems have to be solved.

- First, at certain nodes a multicast packet has to be split into multiple copies in order to reach all destinations, the challenge being to decide when such a copy should be created.
- Second, the recovery strategy used to escape from a local optimum needs to be adapted to take multiple destinations into account.

It is obvious that improvements to unicast routing algorithm like the aid of digital road maps described in chapter 4, par. 3.5, result in improvements to multicast routing, too.

Multicast needs to establish a distribution tree among the nodes, along which packets are forwarded towards the destinations. At the branching points of the tree, copies of the packet are sent along all the branches.
Two – potentially conflicting – properties are desirable for such a distribution tree: (1) the length of the paths to the individual destinations should be minimal and (2) the total number of hops needed to forward the packet to all destinations should be as small as possible. If the topology of the network is known, a distribution tree that optimizes the first criterion can be obtained by combining the shortest paths to the destinations. Wherever these paths diverge, the packet is split. The second criterion is optimized by so-called Steiner trees, which connect source and destinations with the minimum possible number of hops.
However, with position-based routing, routing decisions are based solely on local knowledge, thus neither the shortest paths to all destinations nor (heuristics for) Steiner trees can be used directly.

The authors of [26] propose a solution for a multicast routing based on geographic routing. Their idea is to implement multicast routing exploiting position based routing to avoid both flooding and state maintenance.
They introduced Position Based Multicast routing for mobile Ad Hoc networks (PBM) [26].
PBM uses locally available information to approximate the optima for both properties (1) and (2).
A forwarding node uses information about the positions of the destinations and its own neighbours to determine the next hops that a packet should be forwarded to. This is done to create an algorithm well suited for highly dynamic networks, like a vehicular network is.
PBM does neither require the maintenance of state about a distribution structure nor does it resort to flooding of the data packets. Instead each node that forwards a multicast packet autonomously determines the neighbours that it should forward the packet to. This decision is based on information about the position of the destination nodes, the position of the forwarding node, and the position of the forwarding node's neighbours. It can be regarded as an adaptation of position-based unicast routing schemes such as Greedy-face-Greedy (GFG) and Greedy Perimeter Stateless Routing (GPSR) to multicast routing.

The Position Based Multicast algorithm proposed works under the following assumptions:

- each node knows its own geographic position;
- each node knows the position of all neighbours within transmission range;

these assumptions can be easily satisfied in an environment with position aware nodes, but another condition must hold for PBM to work:

- each node must know the positions of all of the destinations. These positions must be included in the packet or must be available locally by querying a Location Service, which in this case must be an all-to-all service, like DREAM's Location Service.

This assumption means that a multicast source inserts in each multicast packet all the receivers' identities, that it must know all the nodes that have joined a given group and also, if not available locally, must insert in each packet, along with each node's address, each node's position.

PBM does not address problems like scalable distribution of group membership and position information.

Implementing "classical" multicasting - an identity driven multicasting - with an underlying position based routing protocol is not a simple task.

Internet multicast delivers multicast packets and maintains group membership with the aid of the network infrastructure and state maintenance. Existing Ad Hoc solutions like ODMRP do the same by maintaining state in nodes.

Group membership knowledge must be available to nodes explicitly or implicitly through state maintenance.

What state maintenance allows, and this is what Internet and existing Ad Hoc solutions want, is a multicast mechanism in which the sender does not need to know the identities of all of the receiver of a multicast packet nor a protocol in which the sender has to put the identities of all of the receiver in the packet header: this can be a not feasible solution and it is not a general solution for all ad hoc networks, especially for wide networks.

If multicast is based on geographic routing and we don't want to store information about group membership, we have to adopt PBM's solution and insert all receivers in each packet's header.

This is not a general and scalable solution since packets can become infeasible to forward due to the too many destinations; the sender must know all of the receivers of each multicast packet and also must know their geographic positions because the underlying routing mechanism is position based.

4. A brief review of routing operations

Before carrying on with this discussion of multicasting, I would like to subsume and introduce, in a broad manner, some scenarios about position based routing, starting with the unicast case and ending with multicasting.
For the multicast routing I distinguish two main types of multicasting: Geomulticasting and Multicasting. The first can be classified as an "Area Driven" multicast routing, while the second as an "identity driven" multicast routing.
These two types of multicasting arise from two different needs.
The first arises from the need to perform multicast operations inside specific geographic areas; thus, it will be utilized to send packets to some defined recipients that are inside a given ("fixed") geographic area(s).
The second addresses the need for multicast operations where only group membership is of interest.
The last is the classical multicast problem, where what matters is the identity of each single node and the group(s) to which it wants to join. Aforementioned multicast algorithms consider this kind of problem, which is the same problem present in Internet.
Geomulticasting is a kind of multicasting enabled by the underlying geographic routing and required by many applications, which want to send packets in predefined areas and want to reach only specific nodes, rather than performing a Geocasting. Some needs for this type of routing arise from the will to reach specific group of nodes like the followings: *all the police cars in a given area*, *only the pedestrian in a given area*, *only motorcycles and light vehicles in an area* etc.

4.1 Unicast geographic routing

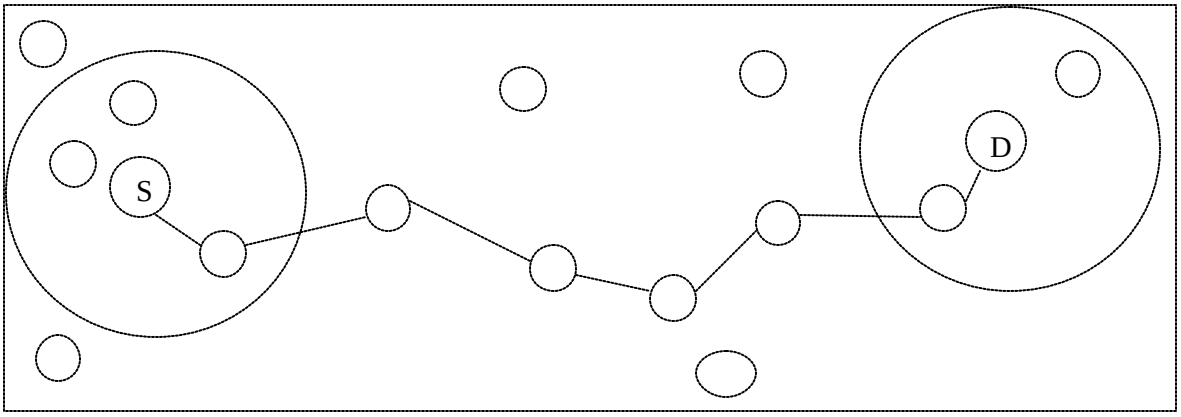


Figure 6.3*. An example of unicast geographic routing.*

A node S wants to send a packet to a node D.

1. With a Location Service S obtains the geographic position of D.
2. S starts the forwarding procedure to send the packet to node D, inserting D's coordinates in the packet's header. If a forwarding node knows a more recent position than the one stored in the packet, then it can change D's position. The forwarding can be done in an efficient manner with "Greedy + Perimeter routing" algorithms, like GFG and GPSR.
   The forwarding procedure can be improved by the use of digital road maps.

4.2 Geocast

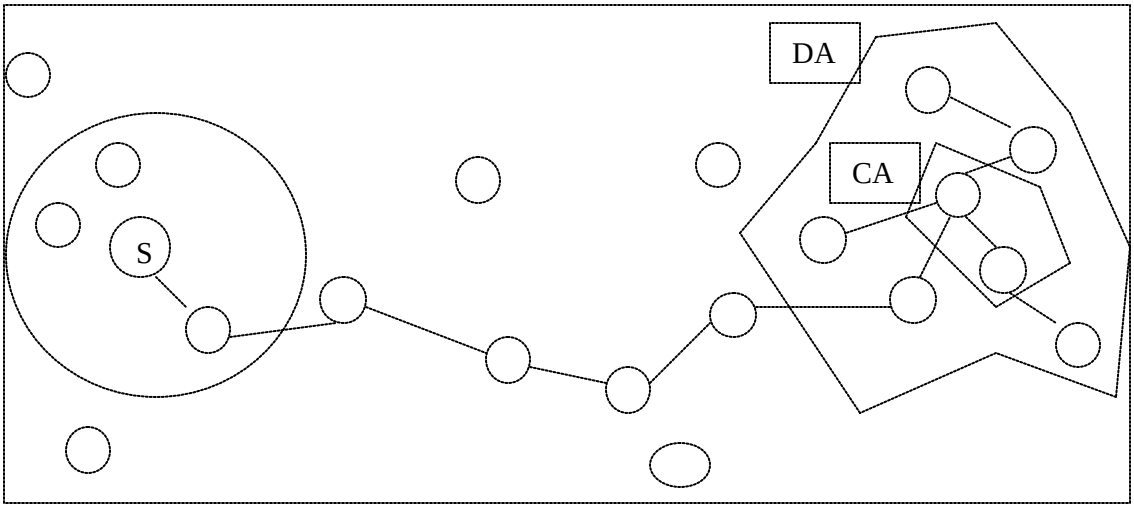


Figure 6.4*. An example of Geocasting toward an area DA.*

A node S wants to send a packet to all nodes inside the destination area DA, specified as a polygon.

1. S sends the packet with destination the polygonal area CA inside the destination area DA (approximately in the centre of DA). The packet can be forwarded with a unicast protocol; this is what previously was called URAD: a modification of a "Greedy+Perimeter routing" algorithm. The first node inside the area CA receiving the packet starts the flooding, controlled flooding or reliable routing for all the nodes inside the destination area DA. A key aspect is the definition of the CA area, for reliability and success.

A location service is of no need.

4.3 Geomulticast

Geomulticast is a sort of "area driven" multicast.
It can be obtained by a Geocast routing in which a *filter* is added while the packet is flooded inside the destination area. Only nodes which have joined the specific group

carried in the packet header process the packet, others only participate in the flooding operation in order to deliver the packet to all the recipients.

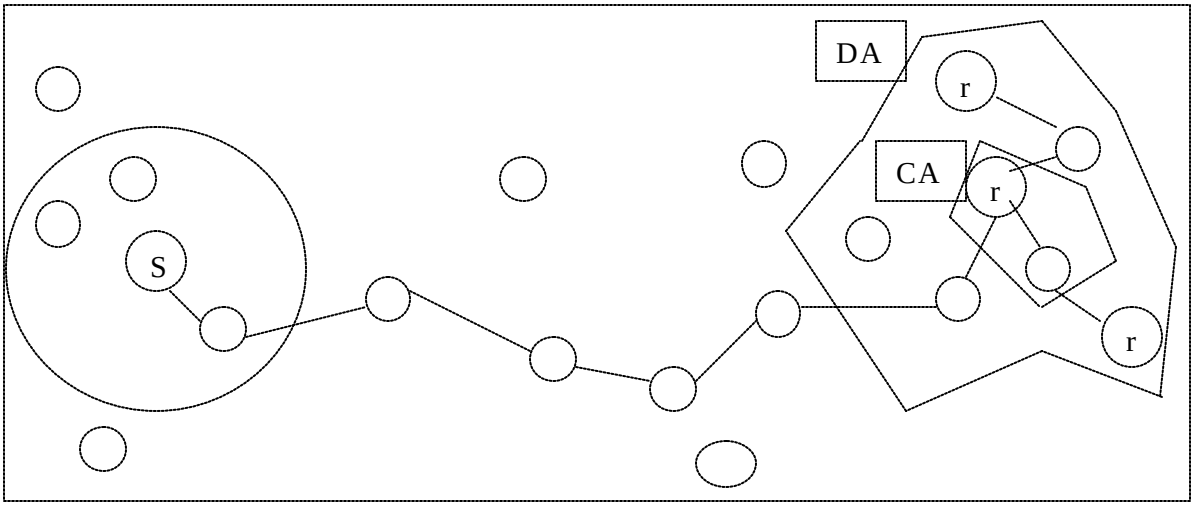


Figure 6.5*. An example of Geomulticasting toward an area DA.*

A node S wants to send a packet to a group of nodes inside a given geographic area DA. What matters to node S is the destination area DA, not the identities of the single recipients: S wants a node to receive the packet if it inside the given area.

1. Node S performs a Geocast operation, but within each packet a given multicast group is specified. A node receives the packet if it is within the destination region and joined the group specified in the packet. A node that did not join the specified multicast group does not process the packet.

With this kind of multicasting no state has to be maintained and no updates are needed.
I made the following assumptions.
A node can join a given group by setting one of some dedicated variables to the value of the desired group. Values must be predefined by an authority, registered, published and updated so that applications are aware of them. Allowing or denying a given application to subscribe a given group must be a higher level task.
The group value is carried in each packet and the *Multicast filter* is performed on this value, other than the geographic position.
That stated, no state, no group's join or leave operations have to be developed in order to have Geomulticasting work in a MANET environment.

4.4 GeoMulticasting toward multiple areas

The following figures show two scenarios in which a Geomulticast operation is utilized to send a packet toward more than one single area.

Example 1:

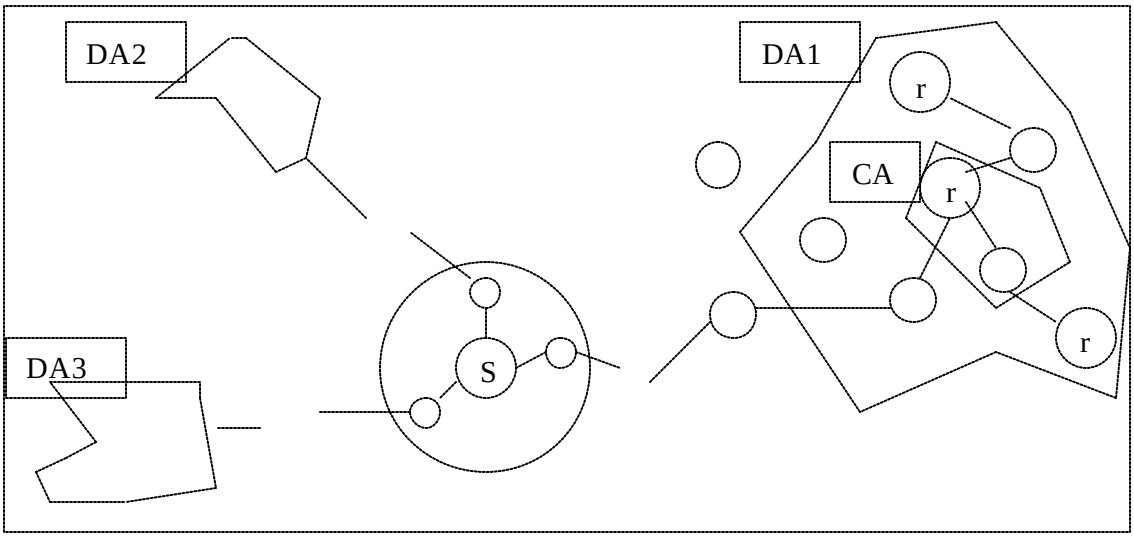


Figure 6.6*. An example of GeoMulticasting toward three different areas. A packet can be split directly by the source of packets S.*

Example 2:

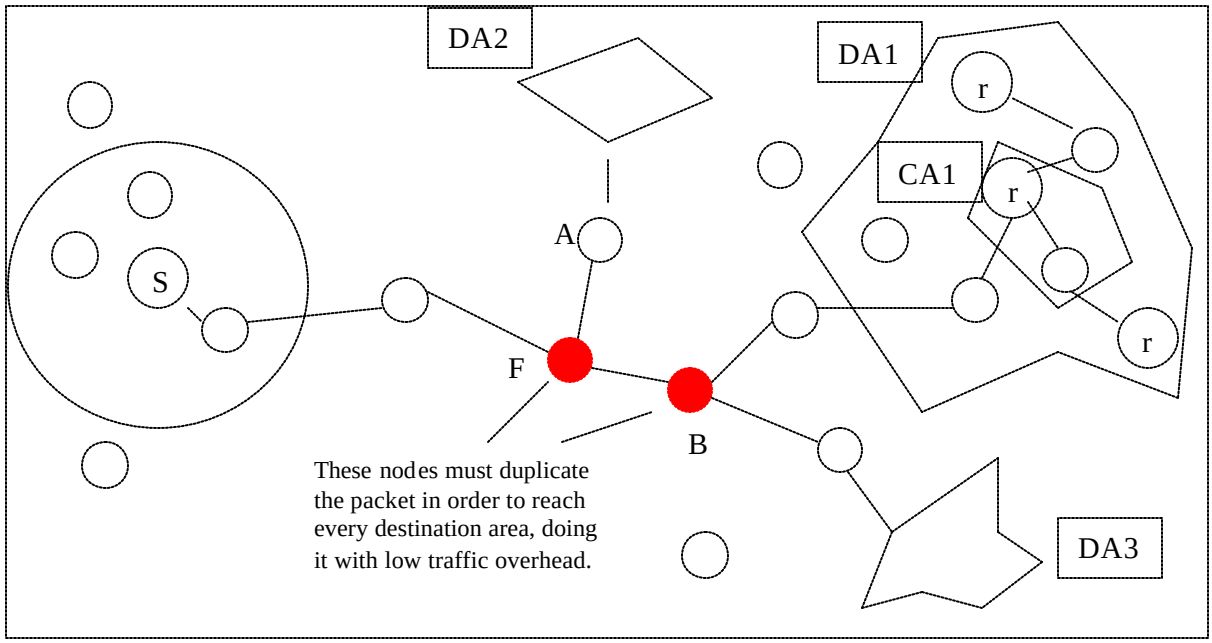


Figure 6.7*. An example of GeoMulticasting toward three different areas.*
A packet must be splitted by intermediate nodes

.

A node S wants to send a multicast packet to three different geographic areas.
Geocasting is utilized to forward the packet, but great attention is required in this case because at certain nodes the multicast packet has to be split into multiple copies in order to reach all the destinations; the challenge is to decide when such a copy should be created.

This case requires that a packet specify all the destination areas; rather than having a fixed destination address filled with a single polygonal area inside the destination region, the packet format should allow inserting more than one polygonal area into the destination address: an array of destination areas.
Furthermore, in order to forward a packet, a host must check each destination and select a forwarding neighbour for each area specified as a receiver by the sender. If only one neighbour is the nearest neighbour to all the destination areas then the packet is sent "as is " to that neighbour.

If distinct neighbours happens to be the nearest node to one or some of the destination areas, then the packet should be splitted and forwarded to those neighbours, which are the best choice in order to forward the packet to the destinations. The node that recognizes the need for splitting the packet should remove from the destination address those areas that it thinks should not be reached by a specific selected neighbour.
For example:
if the source S could send a packet with the following format:

Number of Destination Areas	*array of polygonal destinations*
Multicast group number	other header fields and payload

taken Example 2 (figure 6.7), the packet would have the following values:

3	DA1 DA2 DA3
predefined group number	other header fields and payload

This packet would be forwarded "as is" until the first red node (F) is met. All nodes until F recognize that the nearest neighbour to each of the destination regions is only one.
The first red node (F) recognizes that neighbour A is the nearest node to destination DA2, while the second red node (B) is the nearest neighbour to the other two areas.
The packet that F sends to A is the following:

1	DA2
predefined group number	other header fields and payload

While the packet forwarded to node B is the following one:

2	DA1 DA3
predefined group number	other header fields and payload

Each forwarding can be done with a “Greedy+Perimeter routing” algorithm like GFG or GPSR until each copy of the original packet reaches its specified destination area. The exploited mechanism is the same utilized in multicasting with a single destination area.

5. Considerations about Multicast and multicasting in vehicular ad hoc networks

Internet-like multicasting is an "identity driven" multicast, where the positions of hosts are not considered, but single identities of sparse nodes are hidden under multicast groups. Network's infrastructure and state keeping take care of executing the multicast operations world-wide.
A specific geographic area does not matter, what matters is only reaching all nodes that have joined a specific group, regardless of their geographic position, i.e. everywhere they are: an unconstrained multicasting.
This approach utilized in Internet is the one utilized by the algorithms ODMRP, CAMP and some other tree-based algorithms in MANETs.
Unconstrained multicast is useful, but it is not a solution that works well in environments with unconstrained mobility and no geographic bounds.

On the other hand, the aforementioned Geocasting and Geomulticasting are operations that reflect the *locality* of an ad hoc network.
An ad hoc network is generally geographically bounded, its aim is not to connect all mobile hosts world-wide (in order have such a wide deployment, aid from fixed infrastructure is needed).
Geocasting and Geomulticasting will be very useful operations in ad hoc inter-vehicles communications, as well as in many other ad hoc scenarios. Because of the fixed receivers of a Geomulticast packet (the destination is not a group of sparse individual mobile nodes, but a sender-specified geographic region), this operation is not complex and does not require much more overhead over a Geocast operation. Geomulticasting, beside Geocasting, is what naturally most vehicular and Ad Hoc networks require, because it is often important to reach all or some hosts in a given area, regardless of their identities.
GPS and other position tracking devices enable the implementation of these operations in a natural way and these operations are what is required above all by some applications in ad hoc environments.
Even in Vehicular Ad Hoc Networks (VANETs), that may be very wide and with lots of nodes, the property of locality remains valid because, although the network can spread for long distances, applications will not involve all nodes in the network, but vehicles within a geographic area or a restricted set of neighbouring vehicles. This is especially true for security related applications, whose alert messages are meaningless outside the dangerous areas. Also traffic monitoring applications will be localized, because

although the result is a road's global status, this can be computed by aggregating observations of distinct smaller geographic areas along the road.
With Geomulticasting, sender nodes do not have to be aware of all of the receivers' identities and positions; Geomulticasting does not need a location service neither suffers from high mobility since it is *Area driven*: the destination is one (or more) fixed area(s) and it consists of no more than a filter over a Geocast operation.

However, although most ad hoc applications will take position into account, many other applications will be *identity driven*.
"Classical" Multicasting is a mechanism that has costs in vehicular networks, while Geomulticasting is a natural approach and not heavy to implement for ad hoc networks and MANETs in which position tracking devices are available. Therefore there is not much space left to the research about Geocast and Geomulticast operations.
On the other hand (classical) Multicasting is an open research area and in the near future *more and more applications will require multicast communications support; also VANETs applications will heavily rely on multicasting.*

A Multicasting with QoS guarantees cannot be efficiently adopted in wide highly mobile wireless networks or even small networks with unconstrained mobility of independent nodes.
However Multicasting can be utilized by some *specific* nodes.
Multicasting can be used in mobile environments by "stable" nodes that can form a group. A group of stable nodes guarantees, at a certain degree of probability, *availability* of communicating partners.
This way Multicasting can be exploited also in *"multicast hostile"* environments, i.e. wide environments of mobile nodes, by some specific nodes excluding others from multicast operations.
Present solutions to classical multicasting for MANETs want to solve the *"unrestricted (about participating nodes), (geographically) unbounded, identity driven*" Internet-like multicast problem. It is clear that mobility of hosts and the Ad Hoc topology pose many challenges.
First of all, *due to the mobility of hosts there could be no partners for a multicast session: disconnections and different mobilities of hosts do not allow lasting multicast communications among nodes*.
This means that it is almost useless to develop multicast applications if communications between nodes cannot last throughout a minimum time interval.

The following work identifies a new semantic for multicasting, where identities of nodes are relevant, but also position and mobility are taken into account to guarantee *availability* of hosts (communicating partners) to multicast applications.

Without availability of hosts no QoS guarantees could be provided by a routing protocol and for multicast applications QoS issues are of utmost importance.
The following work describes a middleware which enables applications to exploit Multicasting even in the highly mobile environment of vehicle networking. It identifies and maintains, in a vehicular ad hoc network, a set of nodes capable to sustain multicast communications among every other node of the set and allows low-latency communications.
I call such a set a *Mobile Multicast Group* (MMG). The only nodes involved in multicast routing are the nodes that are member of a MMG.

Chapter 7

LVMM – The *Localized Vehicular Multicast Middleware*

Introduction

Multicast consists in sending a packet from one sender to multiple receivers with a single operation. It allows a sender to distribute data to all interested parties over a network while minimizing the use of network resources.
Multicasting works well when the underlying network is *stable* and *connectivity* is always available. QoS issues are fundamental to multicasting and without these properties QoS cannot be guaranteed.
GeoMulticasting works with a given destination area, thus on hosts in a *fixed geographic area*.
The basic problem to Multicast deployment in MANETs is that neither no fixed area can be defined, like we can do with Geomulticasting, nor we have fixed hosts' positions, like we have in the Internet environment.
In MANETs environments, Multicast mechanisms have to respond to network dynamics in addition to group dynamics.
In order to use Multicast mechanisms even in a highly mobile environment some degree of network *stability* is fundamental.
Without a stable physical topology no multicasting with QoS guarantees is feasible, neither lasting multicast communications among nodes.
Even when the underlying network has an *overall unstable* topology, looking at *geographic proximity* of some hosts over intervals of *time*, a *stable sub-topology* of the whole network can be identified.
Clearly such a sub-topology is mobile, not bounded to a fixed geographic area and what is important (and is fixed) is the identity of each node: this is a form of the classical Multicast problem, an "identity based" multicast, not bounded to a given geographic area because nodes involved are mobile.
Such a stable -topology identifies a group whose nodes can exploit Multicast operations.

In mobile networks and in vehicular networks (VANETs) above all, hosts continuously move and that is why a stable multicast group cannot be identified in a fixed geographic area; but by exploiting *vehicular mobility patterns and hosts proximity*, a *mobile multicast group* for some of the nodes can be identified. Even if, in the overall network topology, this multicast group is continuously moving, changing its geographic coordinates, taken in isolation from the rest of the network, it is a *stable* group in which hosts' movements are similar and those hosts can form a *relative topology* which is *time-stable*.
I will call this topology a *Mobile Multicast Group* (MMG). It is a *time-stable, bounded* group of physically near nodes. The boundaries of this group are not defined by a fixed

geographic area, like in GeoMulticasting, but by a mobile surface with a limited *diameter*.
This way, l*ocalized* multicast applications (applications that are run collaboratively by group of nearby nodes and thus match the ad hoc model best) can be run also in a highly mobile environment (where what are fixed are not the geographic positions of nodes but their relative distances, while they are moving).
Localized applications are run collaboratively by nearby nodes and thus match the ad hoc routing model best.
Mobile nodes may engage in chat, share multimedia files, engage in video-conferencing, play distributed games etc.

The challenge is finding in a *"multicast hostile"* environment a set of nodes suitable to run multicast operations and making multicasting work among those nodes. These nodes must be independent from other extern nodes about multicast support or use them only when strictly needed. Finding such a set of nodes in a mobile environment will enable hosts to utilize a multicast service with QoS guarantees also in a highly mobile network.

In this chapter the terms "vehicle" and "node" are utilized as synonyms.

1. Multicast support for Vehicular Ad Hoc networks

Preliminaries:
- Mobile Multicast Group, a group of mobile nodes that, for a defined probability a, maintain a stable topology for an interval of time of at least t
- Localized Vehicular Multicast Middleware, a middleware enabling distributed low-latency multicast applications to execute in VANETs: it identifies MMGs, defines a multicast protocol and maintains multicast routes among members.

Primarily, the presence of a stable underlying topology is a lack in MANETs environments, but is instead a basic property of the standard Internet. This stable topology, that is world-wide in Internet, allows the building of multicast services with QoS issues.
Without such a topology, there would be no communicating partners, which can interact over a minimum interval of time that applications require in order to have just a meaning to exist. For example, distributed entertainment applications require that interactions between nodes last at least over a minimum interval of time, otherwise it is clear that there would be no reason at all for such applications to run.
But, as aforementioned, also in high speed vehicular ad hoc networks, there are some scenarios where multicast groups can persist throughout quite long time intervals: some mobility properties allow us to identify groups of vehicles that can remain near each other and form a *Mobile Multicast Group*.
Such a *Mobile Multicast Group* (MMG) can be a *driver aware* group, where drivers of different cars remain near each others because they know each other and want to make their way together; more frequent, *implicit on velocity* groups can arise due to similar velocities of nearby vehicles along a roadway.
It can also be envisioned the automated formation of platoon of vehicles running the same multicast application.
In both cases, a MMG represents a multicast-enabling configuration inside VANETs, which several inter-vehicles multicast applications can be based on.

Definition 1: Mobile Multicast Group (MMG).
A Mobile Multicast Group (MMG) identifies a group of mobile nodes that for a defined probability a maintain a stable topology for an interval of time of at least t. It is a time stable, diameter bounded configuration of neighbouring mobile nodes. These nodes show similar mobility patterns and mobility rate and are located in a mobile, diameter-bounded surface.

Each node member of a MMG can be the source of multicast packets; *a MMG is an all-to-all multicast group*: *when a node sends a packet all other members receive it.*

The fundamental characteristic of neighbouring nodes to form a MMG is the similarity of nodes' movements. Vehicles that are members of the group have velocities close to each other's velocity but deviate slightly from it and these velocities characterize group's velocity. MMG's stability is determined with the aid of a mobility prediction algorithm that computes probabilistic bounds on the availability of paths to member nodes over a specified interval of time.
A *Mobile Multicast Group* is also a good basic block for QoS mechanisms as will be explained below.
Thanks to the properties of a MMG, nodes of a such a group can run together a multicast routing protocol and several applications can exploit it: they can run distributed chat on mobile nodes, movies sharing, distributed entertainment applications etc.

Based on the concept of MMG, we introduce a *middleware* layer that, from a highly mobile network, finds a set of nodes suitable and willing to form *Mobile Multicast Groups* (MMGs). These nodes are physically near to each other and it is expected that they will remain near each other for a medium/long interval of time.
This group of nodes runs a low-latency multicast algorithm that is able to deliver messages sent from every node of the group to every other node of the group.
Such a group provides applications with a multicast communication support in the highly mobile environment of Vehicular Ad Hoc Networks (VANETs).
This middleware is called the *Localized Vehicular Multicast Middleware (LVMM).*

Definition 2: Localized Vehicular Multicast Middleware (LVMM).
Localized Vehicular Multicast Middleware (LVMM) is a middleware enabling vehicles to execute low-latency distributed multicast applications in VANETs; it identifies MMGs, defines a multicast routing protocol and maintains multicast routes among members.

LVMM's main function is to build and maintain MMGs and to route multicast packets among groups' participants. In order to maintain its state it uses a *pro-active* approach among the members of the MMGs (and *only* among them).
The frequency of pro-active updates is controlled, from the others, by the mobility prediction algorithm so that it can be reduced, based on network's properties. Given that nodes are mobile, a *pro-active* approach that takes into account nodes' mobility is essential, because the ability to predict the future state of the sub-network of interest is

fundamental, if communication algorithms are expected to maintain any substantive quality-of-service (QoS).

As stated previously, a MMG supports all-to-all multicast communications, i.e. each member can be source of multicast packets.
Applications like distributed chats, video-conferencing, distributed entertainment and, in general, real-time multi-user applications require multicast (all-to-all) mechanisms to have their messages delivered to each node running the application; they can rely on LVMM to deliver messages to other nodes.
Aspects of *QoS* are another major concern. LVMM is built with *QoS* in mind, which is supported with a *pro-active* state maintenance.
LVMM defines an architecture which is suitable for *low-latency* multicasting in a mobile environment by finding a group of nodes suitable to perform multicast operations and by maintaining updated multicast routes.
Members of a MMG maintain an updated routing table utilized to route multicast packets and always know other members. This way, because a proactive protocol is utilized, the routing path followed by each multicast packet is a sequence of nodes that is not explicit: it corresponds to a next hop table lookup at each vehicle along the route.
Packet forwarding is faster than with on-demand protocols: they are not suitable for low-latency packets exchange due to their delay to obtain a route. A MMG is a group of nodes in which packet forwarding is fast because each node has a state that makes the node always aware of the forwardings it must perform in order to optimally route packets (so that they are received by all other members with the least number of transmissions and without duplicated packets).
QoS is a very important aspect for inter-vehicles communication and it cannot leave out of consideration *availability* of communication's partners and *low-latency* routing.
To testify this need, below is reported a phrase extracted from the VANET [74] home page. It is about the scope of the VANET forum:
"*The vision is safety and commercial applications enabled by short to medium range communication systems and/or networks (vehicle-vehicle or vehicle-roadside). Such technology should provide priority for time-critical safety messages and meet the QOS requirements of other mobile e-commerce or multimedia applications.*"

In LVMM, a *robust* and *low-latency* multicasting can be provided to applications. With a global soft state each node is always aware of the identities of all other members: the sender of each packet is known and each sender knows the identities of all of its receivers.
All members of a MMG can be trusted and authenticated by a decentralized protocol before they are allowed to the group.

2. What applications can get

The ultimate goal of LVMM is to assist software developers in their efforts to design and build multicast applications over vehicular ad-hoc networks. The key to LVMM strategy is to provide, at the application level, the appearance of stability in a domain that is characterized by high degrees of mobility. Even though many different factors can contribute to communication failures, we assume that short-lived transient failures are masked by the communication layer, thus relegating all disconnections to the mobility of nodes. In such a setting, the application programmer perceives the configuration (i.e., group membership) to be stable if changes to it are atomic, i.e., cannot affect any operation already in progress.
LVMM has the task to maintain group and multicast routes that applications can transparently utilize to sustain multicast communications. LVMM maintains information about the nodes that belong to a group and delivers the messages to them.
Thus, applications do not need to worry about the underlying network, neither groups' formation nor the routing protocol. This promotes modular design of distributed applications.

Ideally, it is desirable to guarantee that on-going connections and their QoS requirements of are preserved for their entire duration. Unfortunately, this is not possible in a time-varying network environment as connections may fail randomly due to user mobility. A more realistic and practical approach is to provide some form of probabilistic QoS guarantees by keeping connection failures below pre-specified threshold values and by ensuring, with high probability, that a minimum level of availability and bandwidth is always available to ongoing connections.
In a highly dynamic network, groups of nodes can be dynamically found which maintain a relative stable effective topology. The membership in each group changes over time in response to nodes' mobility and is determined by the criteria specified in the algorithm responsible to create and maintain the group. A group is a set of nodes constrained on the number of participants and the diameter of the area covered by group members. These values are parameters that determine groups' characteristics.

Applications through *Localized Vehicular Multicast Middleware*:

- can create/join a multicast groups;
- can transparently send/receive multicast messages with low-latency;
- are aware of the identities of all the members of a group (a service built upon the routing protocol).

3. What LVMM assumes, how LVMM works

The *Localized Vehicular Multicast Middleware* assumes that each vehicle is equipped with:

- A wireless device that allows the vehicle to transmit/receive.
- GPS, D-GPS, or other position tracking devices that allow an instantaneously knowledge of its positions.
- Computational power.

LVMM creates and maintains *Mobile Multicast Groups* (MMGs) among vehicles on a roadway and it allows multicast communications among the members of a MMG.
The definition of a *Mobile Multicast Group* was given above.
LVMM models a *Mobile Multicast Group* in a vehicular ad hoc network as a graph $\mathbf{MMG} = (\mathbf{V},\mathbf{E})$, where V is the set of member vehicles and E is the set of bi-directional communication links among the vehicles.
Such graph is called the *MMG-graph*.
A *MMG-graph* models a MMG in its entirety.
A *MMG-graph* is a sub case of the widely known *unit disc* model: a basic graph-theoretical model for ad hoc networks.
A *unit disc* graph is defined in the following way: two nodes A and B in the network are neighbours (and thus joined by an edge) if the Euclidean distance between their coordinates in the network is at most R, where R is the transmission radius which is equal for all nodes in the network.
A *MMG-graph* changes over time.
The presence of an edge (V_u,V_v) indicates that host V_u is within transmission range of host V_v and vice versa. In practice, each host can make itself known to its neighbours by generating a beacon at regular intervals and by listening to signals from other hosts around. When a beacon ceases to be heard, a node is considered to be no longer within transmission range. The frequency of the beacon transmissions determines the accuracy of the information available at each host.

In the following, the specific nature of a *MMG-graph* will be shown.
Let's consider a bi-dimensional plane with the coordinates x and y.
Let's assume that a roadway develops along the x axis. The example is a MMG made up of five vehicles; it represents a MMG at a given interval of time t. All other vehicles do not matter because they have a different mobility or they are not members of the MMG.

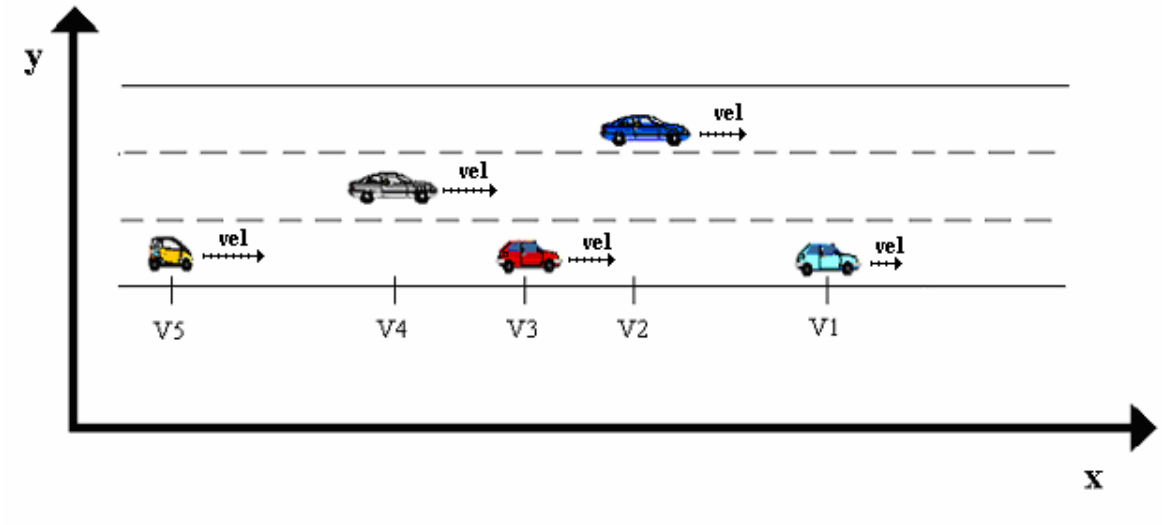


Figure 7.1*. A Mobile Multicast Group made up of five vehicles along a stretch of roadway.*

Every vehicle is represented by the following values at a given node:

- A unique identifier (**ID**)
- Its position, obtained by the GPS device (**Pos**)
- Its velocity (**Vel**)
- A transmission range that is assumed to be the same for all the vehicles (**R**)

Thus, each vertex V of a *MMG-graph* represents a vehicle by identifying it with the above values, that is, each vertex $\mathbf{V}_i$ is described by the following notation:

$$\mathbf{V}i = \{\ \mathbf{ID}_i, \mathbf{Pos}_i, \mathbf{Vel}_i, \mathbf{R}\ \}$$

Each vehicle is identified by a unique ID; ID_i indicates the unique identifier of the node V_i, not the identifier itself. Pos_i and Vel_i respectively identify the position and velocity of the vehicle V_i; the Pos value is the parameter that determines the position of a vehicle in the MMG.
The parameters that identify each vehicle of a MMG, at a given interval of time, are its *ID* and its *Pos*.
The values *Pos* and *Vel* of each node are utilized to build and maintain the MMG because they allow every vehicle to monitor its neighbours. The radius R determines the links among vehicles.
The stretch of roadway containing a MMG can be represented with a straight segment, that is:

- Roadway's width is negligible (y axis)

- Roadway's shape is negligible (a segment parallel to the x axis)

We can omit roadway's width and shape because transmission ranges cover all roadway's width and vehicles utilize wireless radio transmissions.
This allows us to consider all vehicles as lying on a straight line and their GPS devices allow us to order all vehicles on the line.
Each Pos_i value represents a position on the x axis, then we can use the term $\mathbf{x_i}$ as the value of each Pos_i.
For simplicity, I assume that all $\mathbf{x_i}$ are different from each other. If it happens that in a time interval two or more $\mathbf{x_i}$ are equals then we can utilize other values from the GPS devices to order the vehicles or in another way find an order.
Thus, the above shown MMG, at time t, can be modelled in the following way:

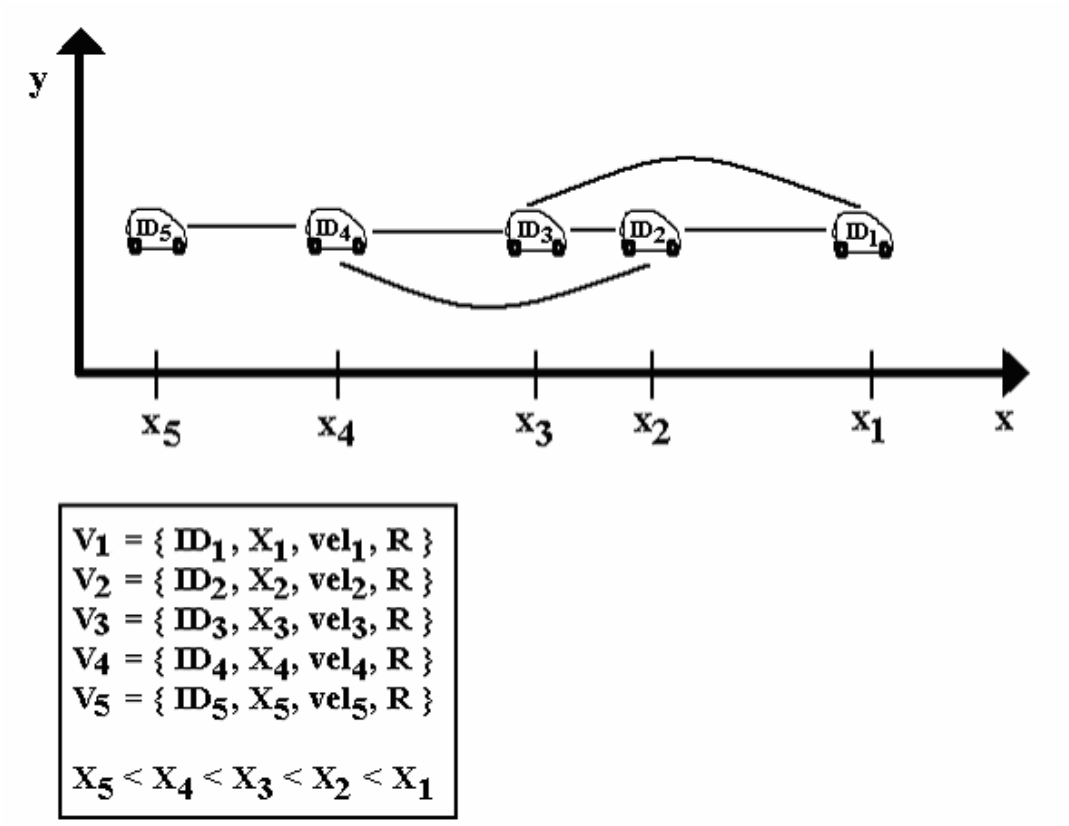


Figure 7.2 *A MMG-graph modelling a Mobile Multicast Group made up of five vehicles.*

For the above graph the set of members is the following:

$$\mathbf{M} = \{\mathbf{ID_1}, \mathbf{ID_2}, \mathbf{ID_3}, \mathbf{ID_4}, \mathbf{ID_5}\}$$

A *MMG-graph*

1. is a *dynamic* graph. A MMG changes over time due to the mobility of its nodes. A *MMG-graph* is valid only for a given interval of time.
2. Is a *one-dimensional* graph. The environment is a stretch of roadway that can be represented with a straight segment. The segment models the given stretch of roadway omitting its width.
3. Is a graph of *ordered nodes*. Each node ID_i has a position X_i that univoquely identifies the node on the segment
4. Models a MMG in its *entirety*. A *MMG-graph* includes all nodes of a MMG and all links existing between each pair of members.
5. It is *bounded* by a maximum segment size (L) and maximum number of members (MaxN). Because a MMG is theoretically unbounded, it is constrained by two parameters: the first, L, is an upper bound on the distance between the leading vehicle and the last vehicle of a MMG; the second, MaxN, is an upper bound on the number of participants. Both are application dependent.

Now a MMG will be defined in a more rigorous way.
Let MMG_t be a MMG graph that models a given MMG in an interval of time t, let N be the number of members of the given MMG_t ($N < MaxN$) and let $\mathbf{M}$ be the set of members of the MMG (they have already joined the group).

$$\mathbf{M} = \{\mathbf{ID} : \mathbf{ID} \in \mathbf{MMG}\}$$

j is a function that given the identifier of a vehicle returns its position, $j : Identifier \rightarrow Position$:

$$j(\mathbf{ID}) = \mathbf{x_i}$$

$\mathbf{X}$ identifies the set of points on the x axis that correspond to the positions of the members of the MMG.

$$\mathbf{X} = \{\mathbf{x} : \exists \mathbf{ID} \in \mathbf{M} \wedge j(\mathbf{ID}) = \mathbf{x}\}$$

(x_i, x_j) identifies a bidirectional link between the vehicle with position x_i and the vehicle with position x_j: it identifies both the unidirectional links $\langle x_i, x_j \rangle$ and $\langle x_j, x_i \rangle$.

For each pair x_i, x_j there exists a link between the correspondent vehicles if and only if the Euclidean distance between them is less than the transmission radius **R**. We assume that each link (x_i, x_j) has the same associated cost.

Let **E** be the set of all bidirectional links that exists between each pair of distinct members.

$$\mathbf{E} = \left\{ (x_i, x_j) : |x_i - x_j| < R,\ x_i \in X,\ x_j \in X,\ (i \neq j) \right\}$$

Then a *MMG-graph* at time t is defined in the following way:

$$\mathbf{MMG_t} = (X, E)$$

A *MMG-graph* always changes with time, thus a $\mathbf{MMG_t} = (X, E)$ represents a MMG in a given interval of time: it is recomputed when lower layers send new updated values about the underlying network.

In this document a protocol is presented that does not require global knowledge of a whole MMG to compute minimum cost routes: each node just has to know about the member nodes of the MMG that it can directly reach. This knowledge is already known at lower levels because it is required to build and maintain group membership.
Each node only needs to know about its neighbourhood, that is, it only has to deal with partial graphs that model its neighbourhood. No node has to pro-actively know about all other members of a group to perform the multicast routing, still the result is a minimum cost multicast tree for each source of packets. Cost is expressed in terms of number of required transmissions.

4. Feasibility

Preliminaries:
- It is feasible to realize a time-stable multicasting on VANETs. Availability of communicating partners can be guaranteed.

Since fixed networks' multicast routing is based on state in routers (either hard or soft), it is judged fundamentally unsuitable for an ad hoc network with *unconstrained mobility*.
The term *unconstrained mobility* implies the following:

- Hosts' behaviours completely independent of other hosts
- No limit on hosts' speed
- No constraints on direction of movement
- High probability of frequent, temporary network partitions

All these factors are valid (although the third less than the others) in vehicular ad hoc networks, too.
Because of all these factors it seems that it is no longer worthwhile for a mobile host to maintain any multicast-related state information other than its own.
However looking at upper points with vehicular traffic topology in mind, we can discover the followings.

- At a first glance the first point appears true: each vehicle is a single independent node making its own way. But if we think at it more deeply, we find that it is not so true, because each vehicle is constrained by other vehicles. Each driver has vehicles in front of him that don't allow him to be totally independent. Sometimes traffic is like a flux along a street. This is true also for highways scenarios. Roads' topologies and rules make each node behaving like other nodes. Differences between vehicles running on the same roadway are only two: first, each vehicle has its own velocity and can increase or decrease it, second, each node can, independently from others, divert its way to another road (this can be like leaving a network).

- Limits on speed exist, but even if they didn't exist other vehicles often force rear vehicles to a speed limit. Think of traffic jams or roads with heavy traffic. This is also true for highways with medium traffic.

- Direction of movement is constrained by roads and directions.

- Network partitions can occur, but many inter-vehicles scenarios are very dense or dense enough to maintain connection. Because vehicles are not power constrained they can adjust their transmitters' powers to reach distant vehicles when they detect the presence of voids around them.

Subsuming, in a mobile ad hoc network a fixed topology does not exists. A stable group of nodes must be found which "extracts", from the set of all nodes of a mobile network on a given segment, nodes that for a defined probability α form a connected (stable) topology for at least an interval of time τ. Such topology was called a *Mobile Multicast Group*.
Such group can be identified and maintained because of mobility patterns along roadways.
Member nodes can run together a multicast algorithm and send messages with QoS guarantees that could not be available in the overall network and they can send those messages with low-latencies.
Maintaining an updated state of the underlying network requires computation and energy powers to hosts. But vehicles are not power constrained and will be equipped with high-power hosts. State maintenance has several advantages in multicasting scenarios; some of them are that every node can be aware of other participants and can immediately forward packets toward their destinations without having to compute routing on demand before forwarding packets and packets do not have to carry destinations in their headers.
Thus, applications requiring low-latency multicast mechanisms can be deployed in vehicular networks and they can rely upon the *Localized Vehicular Multicast Middleware* to have their messages reach all subscribed hosts.
State maintenance of the routing protocol utilized by LVMM is "light" because each node only needs to know about its directly reachable neighbours; still the final result is a global source-based minimum cost multicasting.

5. The building block

Preliminaries:
- Time-Proximity Group; for each node N, the set of vehicles within N's transmission range that are suitable to form a multicast group with N, i.e. nodes that maintain reachability over a defined interval of time.

- Time-Proximity Groups are MMGs' building blocks.
- Mobility prediction is the basic mechanism utilized to identify a Time-Proximity Group.
- Pro-active state maintenance supports mechanisms for availability and QoS guarantees by monitoring the network. The frequency of pro-active updates is regulated by mobility prediction and QoS issues.

Finding and maintaining a set of nodes, which are suitable to run a *time-stable low-latency* multicast protocol among every other node and maintaining the multicast routes among all of the members are the main tasks of the *Localized Vehicular Multicast Middleware* (LVMM).
The consequences of nodes' mobility suggest the need to include a quantitave measure of mobility directly in the nodes' selection process.
The group of stable vehicles is discovered by having each node monitoring all its directly reachable neighbours. This monitoring is performed by analyzing the special beacons that are periodically broadcasted by all vehicles to their neighbours.
Neighbours that maintain velocities close to the velocity of a monitoring node, for a given interval of time, are considered "stable" vehicles by the monitoring node: this way a group of vehicles can be created, with the property that relative distances between members will not show great differences on a short interval of time. The proximity property among vehicles will hold for intervals of time that can be very long, depending on the way taken by nodes and traffic conditions.
The basic building block of a MMG is a mechanism that checks at every node the set of directly reachable vehicles that are suitable to form a multicast group: I refer to this set of directly reachable nodes as the *Time-Proximity Group* (TPG) of a given vehicle.
A *mobility prediction* algorithm is utilized to set up and maintain a TPG and to filter out multicast-unreliable neighbours.

***Definition 3 : Time-Proximity Group* (TPG).**
For each node N, a Time-Proximity Group (TPG) is a set that holds all the nodes inside N's transmission range that will remain within N's proximity (i.e. will be directly reachable) for at least a period of time t with a probability of at least a - TPG(t,a) -.

A TPG is the building block of a MMG (one or more MMGs).

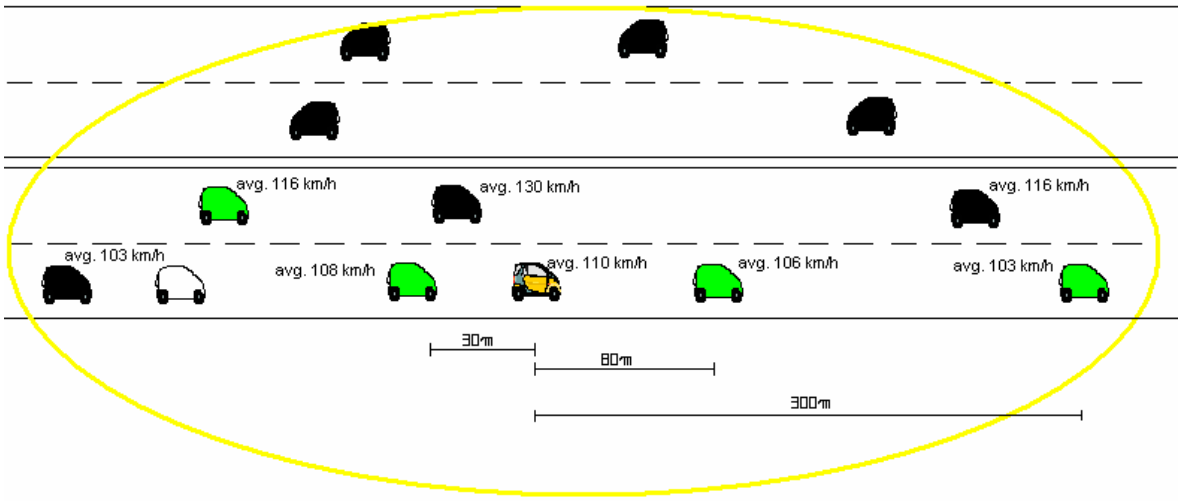


Figure 7.3*. A vehicle and its Time-Proximity Group (TPG) for t = 5 minutes, in a given interval of time.*
Like this figure shows, every node is the centre of a circumference (or an ellipsis if, with directional antennas, it can restrict and direct its radio signal to adapt to roadway's topology) with radius the transmission range of the node.

A *TPG sub-layer* in each node is responsible for creating and maintaining the TPG(s) of the node.
Yellow car's *TPG sub-layer* runs a mobility prediction algorithm, whose parameters are α and τ and it selects vehicles suitable to satisfy those requirements of stability.
The above figure shows a highway segment covered by the yellow car's transmission range. Vehicles inside yellow car's transmission range can become members of its TPG (and have yellow car in their TPG). Black cars are vehicles that are not members of yellow car's TPG. This may happen because

- they have a different mobility: vehicles travelling in the opposite direction can never become members of the TPG (a first removal of these nodes could be

done directly at the radio transceiver level, by adapting the radio propagations along the road along the direction of nodes' movement)

- they have a slightly greater velocity than yellow car but are in front of it, near its radio limit,
- they have a slightly lower velocity than yellow car but are near its radio limit behind it.

A white car is a vehicle that does not support LVMM or simply it is not equipped with wireless networking devices. A green car is a car that is in yellow car's TPG.

A radio range of 300m is shown in figure, but it can be greater and clearly depends on the transmission technology utilized. Technologies such as DSRC and the possibility to adjust transmission power based on different network topologies can enable longer transmission ranges.

Figure 7.3 shows the *Time-Proximity Group* perceived by the yellow car, in a given interval of time, for t = 5 minutes. This TPG represents a stable neighbourhood over an interval of time of at least 5 minutes. In the above example the parameter α is not considered: it is omitted because figure 7.3 represents a topology in a single interval of time and for simplicity the roadway has no diversions, this way, only velocities of vehicles determine variations on TPG's membership and we can omit α and simplify the example.

In a real environment, a multicast application (for example a video-chat or generally an infotainment application) may require, for example, that nodes be always reachable for at least one minute with a probability of 0.70. Then it means that, at each instant, the mobility prediction algorithm at each node will filter out all neighbours that from sample observations will not be judged, with a probability a = 0.70, as directly reachable after an interval of time of at least t = 1 minute.

t is not a parameter that specify the minimum required length of time for a communication, but the minimum length of time of the future stability that is always required to every node: at each instant, only those nodes that will remain stable throughout the successive interval of time, whose length is at least t, with probability a (lower bound), enter the TPG(t,a).

We have that $\mathrm{TPG}(t', a) \subset \mathrm{TPG}(t'', a)$ if $t'' < t'$.

For example, given the same scenario of figure 7.3, we have the following TPG of yellow car for t = 1 minute.

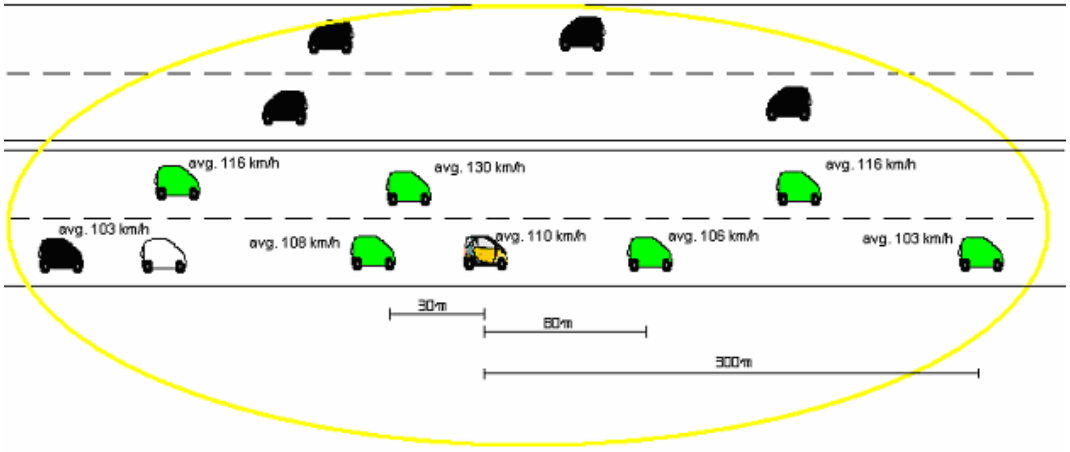


Figure 7.4*. The Time-Proximity Group of yellow car for t = 1 minute, for the same scenario of figure 7.3.*

We have that for the same interval of time $\mathrm{TPG}(5, a) \subset \mathrm{TPG}(1, a)$: all vehicles of the upper lane (of the lower roadway) are green because they will remain within yellow car's transmission radius for at least 1 minute after the depicted time interval.
The figure below again shows yellow car's TPG for t = 1 minute, but 20 seconds later.

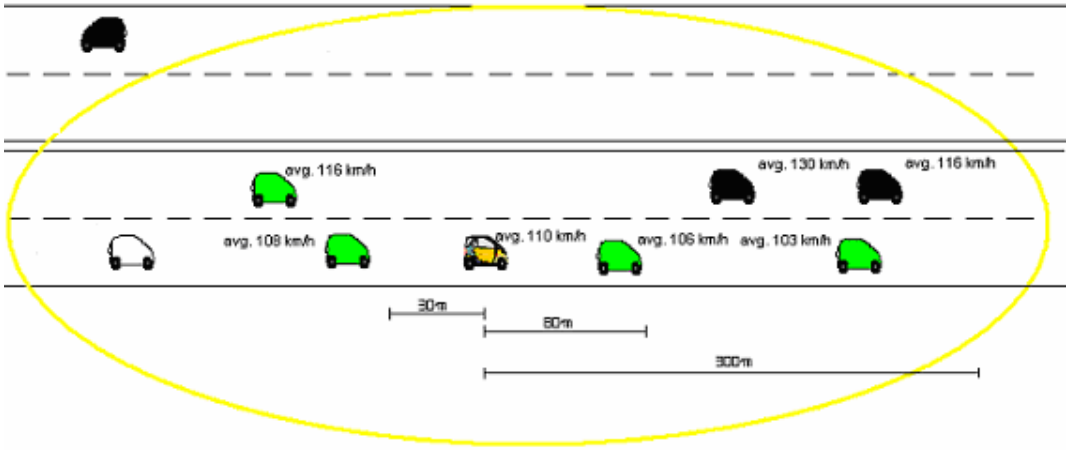


Figure 7.5*. The Time-Proximity Group of yellow car for t = 1 minute, 20 seconds later the scenario depicted in figure 7.4.*

We see in figure 7.5 that the rightmost vehicle with an average speed of 116km/h and the vehicle with an average speed of 130km/h are no more inside yellow car's TPG for t = 1 minute (they are black).

Clearly, the underlying topology must be continuously, i.e. pro-actively, monitored because vehicles may change their velocities or even take a different road.
The middleware, upon request, should notify applications of available connections, the identities of available nodes and a prediction of connections' lasting.
Large values of t tend to result in smaller TPGs: they imply more group stability, however they make it more difficult to achieve the required lower bound a.
Parameter t can be application dependent and its real maximum value is determined by network's topology and mobility. When stability for long intervals of time is required, the middleware can try to provide stability for a time $t < t$ when network topology does not allow reaching better results or when a high value of t would not allows finding partners. With its pro-active state, the middleware will then verify if connectivity can still be maintained after interval t.
Parameter a controls the minimum level of stability of a MMG required to support the traffic load and QoS requirements of connections.

Each TPG is the basis for the *time-stable low-latency* multicasting provided by LVMM, because a TPG contains only those nodes that are selected by the mobility prediction algorithm that satisfy the stability requirements.
Based on mobility prediction the *TPG sub-layer* periodically re-evaluates TPGs' membership to maintain the state up-to-date.
A TPG, thus, holds both proximity and temporal properties on a per-node basis and those properties, when made global to the MMG, determine group's properties. Its name is derived from its properties: Time is for the temporal property, while Proximity is for the physical property.
In a vehicular ad-hoc network a *Time-Proximity Group*, more likely, will arise because neighbouring vehicles have similar velocities and passengers are not aware of this fact (they will notice it if they run an application relying on LVMM).
However in a vehicular network because two or more drivers know each others and make the same trip or simply because they want to continue to run together a multicast application, they may communicate their paths to each others (and maybe they will likely form a platoon of vehicles and continuously communicate with each others).

A *Time-Proximity Group* (TPG) for a node N is made up by all "multicast suitable" nodes that are in the radio range of N. Proximity could be taken on nodes two or more hops away from each node, but this way other nodes should be involved in group

maintenance and in routing of applications' messages; and also the broadcast of beacon packets at the lower level.
Utilizing other non stable nodes would lead to a non robust multicasting, with no QoS basis or even no multicast communications at all.
A MMG is built upon a subset of all nodes in each TPG. Not all nodes in the set of all TPGs are involved in a MMG, but only some selected nodes. This selection is application driven.
LVMM limits MMG propagation by defining a diameter beyond which a group cannot physically extend. This diameter limits the distance that is allowed between the first and the last vehicles and it depends on the maximum sustainable latency for multicast messages.
For the same purposes LVMM can limit the maximum number of members that are allowed to join a given MMG.

6. LVMM at work

The main task of LVMM is the definition and maintenance of MMGs.
Above the *TPG sub-layer* is built an MMG of nodes that run together a distributed multicast application. Only nodes members of a TPG defined by some given parameters a and t, can become members of the MMG defined by the same parameters a and t.
And only nodes that want to receive or send multicast traffic are members of a MMG.
Thus, there are stable nodes that are in the TPGs of other nodes and unstable nodes that are not in any TPGs.
Unstable nodes cannot join a MMG.
Some stable nodes may run a distributed multicast application and thus form a *Mobile Multicast Group*, while others of them may not be interested in a multicast session.
The main task of the *Localized Vehicular Multicast Middleware* is to support vehicles that want to establish a multicast communication and to allow applications to exchange multicast packets without worrying about the underlying unstable topology.
In the following, a simple approach is considered to describe LVMM: only vehicles that run a multicast application P can become member of the MMG for the application P. The real approach utilized by LVMM is the one described in section 9 of this chapter. Because the two approaches have equal architecture and protocols, we utilize the simple approach to describe LVMM. The main difference among the two approaches is about MMG membership: with the simple approach only nodes that run a multicast application join the MMG associated to the application, while with the other, nodes not interested in multicast packets can be member of a MMG and provide a richer connectivity. No difference exists about architecture and routing protocol between the two approaches.

The simple approach utilized by LVMM is enabled by the peculiarities and properties of a VANET, like the high power of transmissions and the underlying network topology, which is determined by roads and by the characteristics of vehicular traffic on the roads. Only the nodes that want to establish multicast communications are involved in the forwarding of multicast packets and in the maintenance of a MMG, without the help of other nodes. Only member nodes, that is, vehicles that want to exchange multicast traffic, receive packets and forward them to other members and maintain the pro-active state for a low-latency multicasting; non-member nodes are not burdened with multicast traffic (it is likely that multicast packets will contain real-time data and real-time traffic always come with its heavy load) and also they are not required to help maintaining the MMG's connectivity updated.

This approach requires that each node that wants to join a MMG *H* for an application P directly reach at least one vehicle that is member of *H* (which run the multicast application P) and, of course, have that node in its TPG.

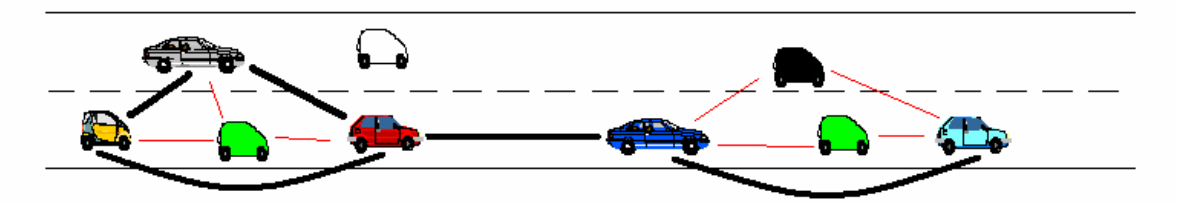


Figure 7.6*. A Mobile Multicast Group enabling vehicles to exchange multicast packets.*

In the above figure, coloured vehicles are members of the same MMG and run the same multicast application. A tick line represents the direct link that allows two vehicles to exchange packets. If a vehicle has a direct black link with another vehicle, it means that the two vehicles are respectively in the TPG of each other and both are members of the same MMG: they can exchange multicast packets with the aid of LVMM. A green car represents vehicles that are in the TPG of other vehicles, but have not joined the MMG; thus, such vehicles will neither receive multicast packets nor will have to maintain the multicast state for other vehicles.
The white car represents vehicles that are not equipped with wireless devices or do not support LVMM.
A MMG is a multi-hop group; a packet sent by anyone of the coloured vehicles will be received by all other coloured vehicles, involving only group members in this process.
A MMG is characterized by the following:
Property 1: only nodes that have in their TPG at least a member, say A, of a *Mobile Multicast Group* can join that group.
If, for a given node N, such a neighbour A does not exists, then N cannot join the desired MMG. If node A exists and it is the only member of the MMG in N's TPG, then A is an '*anchor*' for N to the MMG, because it is the vehicle that enables N to send/receive multicast messages.

7. LVMM explained through its layered architecture

A complete architecture for the introduced *Localized Vehicular Multicast Middleware* is shown below. It addresses all relevant aspects from the physical network to the multicast protocol definition.
The proposed architecture is made up of 2 core sub-layers: the *TPG sub-layer* and the *MMG sub-layer*. These two sub-layers address the tasks of network monitoring, multicast groups' maintenance and multicast protocol definition.
The functionalities, the relevant information and the specific role of each layer will be explained below.

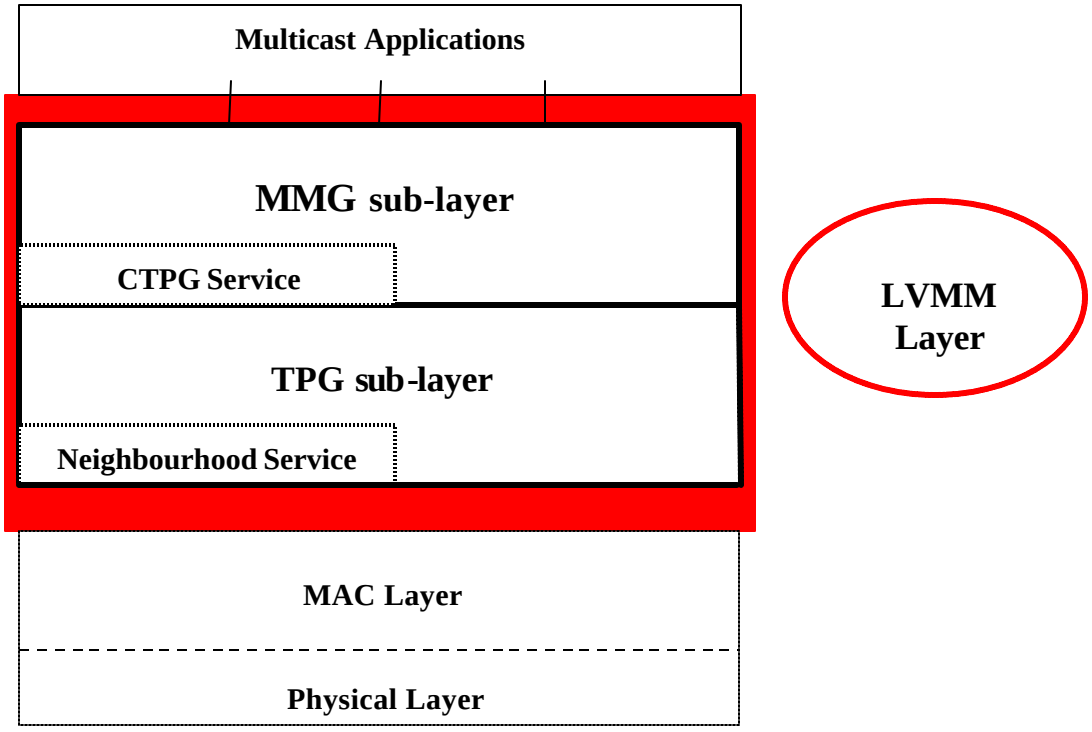


Figure 7.7. *The layers that compose the architecture of LVMM.*

7.1 The MAC Layer and Physical Layer

The Physical Layer provides the physical links among vehicles. It is responsible for frequency selection, carrier frequency generation, signal detection and modulation.
The MAC Layer is responsible for the multiplexing of data streams, data frame detection, medium access and error control. It ensures reliable point-to-point and point-to-multipoint connections. The MAC Layer establishes communication links for data transfer and must fairly and efficiently share communication resources between nodes.
Principal candidate for both Physical and MAC layers is DSRC (chapter 2), although 3G cellular technologies like UtraTDD [76] are likely.

7.2 The Neighbourhood Service

The *Neighbourhood Service* monitors changes in the network configuration within the receiving range of a node. It works by utilizing the medium access control (MAC) layer communication. At this level each node periodically sends and receives beacon messages to let other neighbours know about its presence and to know about other nodes. The *Neighbourhood Service* provides the *TPG sub-layer* with this information. The frequency of the beacon transmissions determines the accuracy of the information available at each vehicle.

In the context of mobile environments, several aspects of group communications must be taken into account.
First, group membership is affected not only by the state of processes (operational or crashed) and links (connected or disconnected), but also by the location of mobile nodes. That's why in a multi-level architecture for mobile systems a *Neighbourhood Service* must reside between the group membership layer and the underlying mobile network.
Second, given a node N, it is the job of the *Neighbourhood Service* to determine the nodes in the vicinity of node N. The *Neighbourhood Service* recognizes all the vehicles within the receiver's range of N. For this purpose neighbourhood messages (beacons) may have to be location stamped as well as time stamped.

The content of a beacon message is the followings:

- Identifier of the vehicle that sends the beacon (**ID**)
- Position of the vehicle (**Pos**)
- Velocity of the vehicle (**Vel**)
- Time stamp (**T**)

These are the values exchanged because each vehicle V_i is characterized by the above data and it is represented by the following 4-tuple in a *MMG-graph*, as explained in paragraph 3 of this chapter:

$$\mathbf{V}_i = \{ \mathbf{ID}_i, \mathbf{Pos}_i, \mathbf{Vel}_i, \mathbf{T}_i \}$$

The Neighbourhood Service at each vehicle collects beacons and records all the neighbouring nodes at real time.
This level uses a list, *NeighboursList*, which contains all directly reachable nodes. Each entry of this list contains each neighbour's identity along with the data received with the beacon messages: its position, its velocity and a timestamp.
NeighboursList is continuously updated so that it always reflects the actual neighbourhood.
This list is available to the TPG layer that through the mobility prediction algorithm finds the subset of nodes suitable to sustain a multicast communication.
Let V_m be the monitoring vehicle; R is the transmission radius of every vehicle.
The *NeighboursList* of the monitoring node can be defined as follows:

$$\mathbf{NeighboursList}(\mathbf{V_m}) = \left\{ \mathbf{V_i} : \left| \mathbf{Pos_m} - \mathbf{Pos_i} \right| < \mathbf{R} \right\}$$

7.3 The TPG sub-layer

The *TPG sub-layer* applies a filter on the neighbouring nodes given by the *Neighbourhood Service* and it presents to its upper sub-layer the sets of vehicles suitable for a time stable communication.
The *TPG sub-layer* utilizes a *mobility prediction* mechanism to select neighbouring nodes suitable to form *Time-Proximity Groups*; it is the base for the *MMG sub-layer* and thus for the creation and maintenance of *Mobile Multicast Groups* (MMGs).
Filtering neighbouring nodes utilizing mobility prediction is the base for LVMM to work.

The *TPG sub-layer* works on the following input data:

- the *NeighboursList* from the lower *Neighbourhood Service*,
- two values: τ and α from the upper layers (they will be specified by applications)

The *TPG sub-layer* utilizes a *mobility prediction* algorithm to find the set of neighbours suitable for a stable communication (chosen from the *NeighboursList* of the *Neighbourhood Service*). These nodes are filtered by the mobility prediction algorithm by utilizing the two parameters α and τ.
The *TPG sub-layer* keeps updated the set of all stable neighbours for each given pair (τ, α): this set of vehicles is the *Time-Proximity Group* for the values τ and α - TPG(τ, α).

Let V_m be the monitoring vehicle. The notation TPG(V_m, τ, α) indicates the specific TPG(τ, α) of the vehicle V_m.
The *TPG sub-layer* at each node V_m proactively updates a TGP(V_m, τ, α) based on the changes to the *NeighboursList*; this way a TPG always reflects the actual topology; it can be defined in the following way:

$$\mathbf{TPG}\left(\mathbf{V_m}, t, a\right) = mobility\_prediction\left(t, a, NeighboursList\left(\mathbf{V_m}\right)\right)$$

The Mobility prediction algorithm

The TGP sub-layer utilizes prediction of node mobility as criteria for localized group organization. The ability to predict the future state of an ad hoc network comprised of highly mobile nodes is essential if the network control algorithms are expected to

maintain any substantive quality of service (QoS) guarantees to low-latency connections. Prediction of node mobility is based on both the velocity of each neighbouring node and its distance from the current node: these values are taken as input from the *Neighbourhood Service*.
The mobility prediction algorithm also utilizes the two parameters α and τ; these two parameters come from the Application layer and identify the degree of stability that an application requires: generally, a multicast application P requires stability of communicating partners for a time of *at least* τ and accepts nodes that have a probability of *at least* α to remain stable throughout time τ.
In the following, we suppose that the roadway has no diversions so that all vehicles follow the same straight roadway. This allows us to omit α and then to introduce a simple mobility prediction algorithm to clarify which actions must be performed at this sub-layer. If the roadway in front of the vehicles had some diversions, then a digital road map would help to estimate the probability that a vehicle could deviate and then leave the neighbourhood. In presence of diversions, a communication of the journey between vehicles would help to better estimate α.
By considering a roadway with no diversions, the mobility prediction algorithm must only take into account vehicles velocities' variations.

Let V_m be the monitoring vehicle and V_j a vehicle in the *NeighbourList* of V_m.
The following is a simple mobility prediction algorithm: its duty is to decide if a vehicle V_j will remain within V_m's transmission range (R) for at least a time τ.
Each vehicle V_i keeps updated a weighted average of its velocity; such weighted average is called H_i. H_i puts more weight on recent samples than on old samples, as the more recent samples better reflect the current vehicle's behaviour.
The simple mobility prediction algorithm is shown below.

```
1.  mobility_prediction(t, NeighboursList(V_m)){
2.    ∀ V_i ∈ NeighboursList(V_m){
3.      if (ID_i ∈ TPG(V_m, t, a)){
4.        H_i = (1−a)·H_i + a·Vel_i;
5.        if (!stable(H_i, Pos_i, t))
6.          TPG(V_m, t, a).remove(ID_i);
7.      }
8.      else{
9.        ask V_i its H_i;
10.       if (stable(H_i, Pos_i, t))
11.         TPG(V_m, t, a).add(ID_i);
12.     }
13.   }
14. }

15. bool stable(H_i, Pos_i, t){
16.   given H_i, Pos_i, H_m, Pos_m and t
17.   if ID_i will remain within ID_m's radio range R for at least a time t
18.     return true;
19.   else
20.     return false;
21. }
```

The above algorithm takes into consideration each neighbour of the monitoring node V_m. If a neighbour J is not inside V_m's TPG (line 8) then the monitoring node asks J a weighted average of its velocity (line 9). By utilizing such average, if V_m determines that J is a stable vehicle, it inserts J in its TPG and records J's average velocity (lines 10, 11).

The method $stable(H_i, Pos_i, t)$ verifies if a vehicle with an average velocity H_i and position Pos_i can remain within V_m's transmission radius R for at least a time τ.

If a neighbour J is already in the TPG of the monitoring node, then V_m updates the average velocity of J with the last communicated speed (Vel) received within the last beacon from vehicle J (line 4). Line 4 utilizes an average called *exponential weighted moving average (EWMA)* [60]; a is the parameter that determines how such average is computed; it can assume small values like 0.125 (i.e. $1/8$).

The use of an *EWMA* is a way to put more weight on recent samples than on old samples to reflect the current state of the network: the weight of a given Vel_i decays exponentially fast as the updates proceed.

After having updated the average speed of the vehicle J, the algorithm estimates if J is still a stable vehicle; if it is no more judged stable it is removed from the TPG and its associated average is discarded (lines 5, 6).

8. The MMG sub-layer

8.1 Mobile Multicast Group - MMG

We have seen that a MMG is a group of mobile nodes that for a defined probability α maintain a stable topology for an interval of time of at least τ; each node member of a MMG can be the source of multicast packets and when a node sends a packet all other members receive it.
Each multicast application P has its own MMG, referred to as MMG(P).
A node can be member of more than one MMG.
A MMG has two parameters that limit its extension:

- MaxN, the maximum number of vehicles that can join a given MMG;
- L, the maximum segment that a MMG can cover in its spread.

Hypothetically a MMG can spread without any restriction until there exist vehicles that can and want to join the MMG. The above parameters limit its spread in two different ways.
The first parameter gives an upper bound to the number of allowable members: this value depends above all on applications and can also depend on latency issues, that is, too many members may waste bandwidth if they all want to be sources of multicast traffic. However if new members only act as receivers, this MaxN can be relaxed, but also this behaviour is application dependent: it is likely that applications will fix an upper bound to the participants and won't allow new nodes to run the same distributed application. For example, a distributed game may sustain up to 20 players, so that 20 is the value of MaxN and no more than 20 nodes are allowed to that application's MMG.
The second parameter L limits the stretch of roadway that a MMG can cover: it holds the value of the maximum allowable diameter of a *Mobile Multicast Group*. Its name stands for "Latency", because by limiting the spread of a MMG we give a limit to the maximum latency that we accept on the delivery of packets.
Every node receives packets sent from every other node, so that a packet sent from one extreme node must reach the opposite extreme node and, if the distance between them is too large, then QoS is wasted and packets forwarding incurs too much latency.

It is clear that *MaxN* and *L* depend on the requirements of applications that utilize LVMM.

For example, the number of participants will be low for video-conferencing applications but for other applications it may be very high.
For example, let's take a highway scenario with a traffic jam. An application is developed that sends traffic information to all the nodes in the traffic jam: it collects and elaborates data coming from the leading vehicles and sends information to the vehicles in the traffic jam.
Provisioning of traffic jam lasting, causes of traffic jam and general traffic information can be valuable to driver inside the jam. Short messages can carry all these data and heavy interaction between vehicles is not required. That is why the application can involve some or all nodes in the traffic jam even for a long stretch of roadway. It will be sufficient that nodes that want to receive real-time information simply execute the application: it will rely on LVMM to join the group for traffic jam news and will immediately deliver those news to the driver.

8.2 Overview of the tasks of the MMG sub-layer

The followings are the main tasks that the *MMG sub-layer* fulfils.

- This sub-layer is responsible for the creation and maintenance of MMGs.
  It periodically broadcasts special purpose packets to let neighbouring vehicles know that a MMG is present in the area so that new nodes can join an existent MMG.
- The *MMG sub-layer* of each node is responsible for the maintenance of the multicast routing tables and the execution of the multicast routing.
  It receives updated values about the underlying local topology by the lower sub-layer and with these values it updates the routing tables of the node.
  When a node receives a multicast packet, this is processed by the *MMG sub-layer*, which forwards the packet utilizing the updated routing table correspondent to the MMG to which the packet belongs. These are the tasks of VMRP, the *Vehicular Multicast Routing Protocol.*
- At this sub-layer each vehicle knows the identities of all other members of a given *Mobile Multicast Group*. Each vehicle maintains a view of the global membership with a "soft state": a table contains an entry for each member of a given MMG and a timeout is associated to each member. If a vehicle does not refresh its membership, with a special purpose multicast packet (*Refresh Membership*), it is no more considered a member by the other nodes when the timeout expires. The "soft state" mechanism relies on the underlying multicast routing protocol to exchange *Refresh Membership* messages.

8.3 CTPG Service

The *CTPG Service* applies a second filter over the neighbouring nodes by selecting from the set of nodes contained in a given TPG only those nodes with which multicast communications are established.
The instance of an application P on a node V_m that wants to establish a MMG(P) (where MMG(P) identifies the *Mobile Multicast Group* for the application P) with parameters α and τ will not generally include in the group all the nodes that are contained in the TPG(V_m, τ, α): only a subset of those nodes will want to run the application.
The set of the stable neighbours of V_m that want to run the application P and thus join the MMG(P) is the CTPG(V_m, τ, α, MMG(P)) of V_m for MMG(P).

Definition 4 : Chosen Time-Proximity Group (CTPG).
Let MMG(P) be the Mobile Multicast Group for an application P and TPG(t, a) the Time-Proximity Group of each vehicle relative to the MMG(P). Each member of MMG(P) has a Chosen Time-Proximity Group (CTPG(t, a , MMG(P)) that holds the nodes contained in its TPG(t, a) that have joined the MMG(P).

A CTPG(V_m, *t, a*, MMG(P)) can be described by the following notation:

$$\mathbf{CTPG}(\mathbf{V_m}, t, a, \mathbf{MMG}(\mathbf{P})) = \{\mathbf{Vi} : \mathbf{Vi} \in \mathbf{TPG}(\mathbf{V_m}, t, a) \wedge \mathbf{ID_i} \in \mathbf{MMG}(\mathbf{P})\}$$

A vehicle belongs to a given MMG(P) if it has successfully passed the *Admission Request* phase; this is represented with notation $ID_i \in$ MMG(P), which will be discussed in the next paragraph.
This means that the vehicle V_m has in its CTPG - defined for parameters α and τ and application P - vehicles from the TPG defined for the same parameters α and τ, which have joined the MMG for application P.
In other words, for each vehicle, a CTPG is built upon a TPG and is a set containing the stable neighbours from the TPG that have joined a given MMG. The subset of the stable nodes that run an application P and join the MMG for that application is given by the union of the CTPGs of the members of the MMG.

It is important to note that a TPG(V_m, τ, α) in unique: it contains the neighbours suitable for a communication with the requirements of stability specified by the parameters α and τ.

On a node V_m, the first time that a running application requires a TPG(V_m, τ, α), that TPG is created and then kept updated till there exists a running application that requires a MMG for the parameters τ and α.
A TPG is application independent: it depends on parameters τ and α.
On the other hand, each application P has its own *Mobile Multicast Group* – MMG(P) and, on each node, its own CTPG. A CTPG is application dependent; it is specific to a given application: if two different applications, say P and Q, want to establish a multicast communication with the same level of stability (the same τ and α), the TPG is only one: TPG(V_m, τ, α). However the two applications have two distinct CTPGs: P has CTPG(V_m, τ, α, MMG(P)) that contains all neighbours of V_m that run the application P (they have joined the same MMG(P)), while Q has CTPG(V_m, τ, α, MMG(Q)) that contains all neighbours that run an instance of the application Q.
The *CTPG Service* does nothing more than recording which stable neighbouring nodes have joined a given MMG. A CTPG is always recomputed with the new values from the underlying network, obtained through the *TPG sub-layer*; identities of vehicles stay the same as long as there are no new admissions or nodes leave the group; the parameters that represent those vehicles are updated.
The identifier of a MMG (MMG(P)) links together a CTPG(V_m, τ, α, MMG(P)) to the correspondent MMG.
The pair (τ, α) links a CTPG(V_m, τ, α, P) to the correspondent TPG(V_m, τ, α).

8.4 How vehicles join a MMG: the *Admission Request* phase

This paragraph explains how a *Mobile Multicast Group* is created and new nodes enter the group.
The *MMG sub-layer* runs the routing protocol, sends/receives global membership messages to maintain global views of the groups and periodically broadcasts advertising messages (*Advertise* messages). The *MMG sub-layer* at each node periodically broadcasts one advertising message for each MMG of which it is a member: this way neighbouring nodes that are not member of a MMG are able to know about the existence of a given MMG and then they can decide to join it.
This way, when a new node enters a 'coverage area' of a group, it is provided with information about the group and an opportunity to join. The group's information should be available as long as there is any member of a group present in the proximity.
Before a new node joins a given group, other nodes, basing on the application they run, decide together if the new node is allowed to the group. This way, a new node entering the group must ask for admission to the group and the group starts an *Admission Request* phase.

When a vehicle ***I*** executes a multicast application **P**, the *Localized Vehicular Multicast Middleware* is invoked and the application sends the parameters α and τ to the middleware, along with the identifier of the application.
LVMM finds the set of stable nodes that have the required mobility established by α and τ. That set is the *Time-Proximity Group* - TPG(τ, α).
Two cases are likely:

- an instance of the application **P** is not already being running by one of the stable nodes. Thus, vehicle ***I*** is the Initiator of the MMG(P).
  In this case, ***I*** creates the group and the nodes in the TPG(τ, α) of ***I*** receive *Advertise* messages and know that a MMG for the application **P** exists.
  The neighbours that join the MMG enter the CTPG(τ, α, P) of ***I*** and ***I*** enters the CTPG(τ, α, P) of those vehicles, too.
  Let *W* and *Q* be the neighbours that have joined the MMG(P). Then the MMG(P) now holds ***I***, ***W*** and ***Q***. From this point, each node that is in the TPG(τ, α) of ***I***, of ***W*** or of ***Q*** can potentally join the MMG(P) and the group can spread along the roadway as long as there are stable nodes and the bounds MaxN and L are not violated.

- An instance of the application **P** is already being running by one or more of the stable neighbouring nodes; say ***T*** one of them. Then the node ***I*** asks to participate to the MMG(P); it sends an *Admission request* message to its neighbour ***T*** to ask if it can join the MMG(P).
  If the nodes member of the MMG(P) agree, then the node ***I*** joins the group. It will update its CTPG(τ, α, P) with all of its neighbours that participate to MMG(P) and will soon receive multicast messages from all other nodes to create its global view of the group. This is because all members of MMG(P) after the *Admission Request* phase have inserted ***I*** in their view of the MMG and the ones that are neighbouring nodes of ***I*** have inserted ***I*** in their CTPG(τ, α, P).

When a vehicle wants to join a given MMG it always asks to be allowed to the group by sending an *Admission request* message to one of the members of the group.
The member that receives an *Admission request* message from a vehicle ***I***, in concert with all other members of the group, decides if ***I*** is allowed to enter the group.
All the members of the MMG run an *Admission Control* protocol to decide if the new node can join the MMG.
The phase from the request of admission until the final positive or negative response of the *Admission Control* protocol is defined the *Admission Request* phase.
Let's assume that a vehicle ***I*** wants to join a given MMG(P).
Let ***T*** be a vehicle in ***I***'s TPG(τ, α), which is member of MMG(P).
I sends to ***T*** an *Admission request* message. The vehicle ***T*** will have ***I*** in its own TPG(τ, α).
The vehicle ***T*** upon receiving the *Admission request* message from ***I***, sends a multicast *Admission Query* packet into the MMG(P). Each member of the MMG(P), upon receiving the *Admission Query* packet, responds to ***T*** with a packet containing a positive or negative response to the new admission. ***T*** collects all the responses and, by utilizing a given policy, decides if ***I*** can enter the MMG. The protocol takes into account concurrent requests of admission with timestamps, so that the limits imposed by the parameters MaxN and L to a group are never exceeded. If ***I*** can enter the MMG then ***T*** sends a packet to ***I*** communicating the positive response. ***T*** will insert ***I*** in its CTPG(τ, α, P).
Because each member of the MMG responded to ***T*** with a multicast message, all the nodes of the group can compute the same result than ***T*** does; then each of them independently from the others can track a List of all the legitimate members.
Each member knows that ***I*** is a new member and the vehicles that have ***I*** in their TPG(τ, α) insert ***I*** in their CTPG(τ, α, P).
I will soon receive multicast messages about all other nodes to create its view of the global group and it will receive application layer multicast packets for the application P.

I will be able to send packets to all the members of *Mobile Multicast Group* for the application P.
I will insert the nodes in its TPG(τ, α) that are members of MMG(P) in its CTPG(τ, α, P).
As will be shown below by explaining the multicast algorithm, it is *necessary and sufficient* that neighbouring nodes that are members of the same MMG insert each other in their respective CTPGs in order to perform the multicast routing algorithm.

If ***I*** is not allowed to the MMG then it will receive a negative message from ***T***. All the nodes of the MMG know that ***I*** has not been admitted. Neighbours of ***I*** will not insert ***I*** in their CTPG(τ, α, P). This way ***I*** will neither receive multicast packets, nor it will be able to send multicast packets.

Admission of new nodes is decided at the application layer: each instance of an application at every node "votes" for the admission or for the refusal of admission of a new node.
The application may leave this decision to the user. For example an application for video-messaging or a distributed game may ask the user if she wants the new vehicle to join (that is, if she wants another user to participate to the application). This way, the positive or negative responses come directly from the users.
If the application accept every new node, than no "votes" has to be collected and transparently the new nodes are allowed to join a group.

Anyway, a node is not allowed to join a group if the group has already its maximum number of hosts or if including the node (MaxN) would make the group cover a stretch of roadway longer than L.

8.5 The Vehicular Multicast Routing Protocol - VMRP

We have seen in section 3 of this chapter how a *MMG-graph* $\mathbf{MMG_t} = (\mathbf{X}, \mathbf{E})$ models a *Mobile Multicast Group* at time t.
This paragraph shows how such a model can be exploited to perform an efficient multicast routing among all the members of a MMG. It will also be shown that each node only needs to know about its neighbours, i.e. its CTPG, without the need of a complete knowledge of the *MMG-graph*.
A *MMG-graph* is a general graph that models a MMG in its entirety, hence, in order to build a *MMG-graph*, each node should have updated information about all other members: its maintenance requires an all-to-all approach that allows each node to get identities and positions about all other nodes. With such data each node could compute a global view of the group and with this global knowledge, each node could, independently from each other, compute the routing paths.
Altough adopting an all-to-all approach is possible, it would introduce overhead because each node should pro-actively maintain an updated view about all other members. This solution is thus not scalable.
In our approach a global knowledge of the MMG is not needed as no node utilizes a *MMG-graph* to route packets. The all-to-all approach is avoided so that each node only makes routing decisions with the data in its CTPG. Still the final result is a least-cost source-based multicast routing: the routing protocol utilizes only local knowledge at each node but

- delivers each packet along a tree rooted at the source of the multicast packet and
- such tree is of minimum cost, that is, it utilizes the minimum number of transmissions.

Avoiding the all-to-all approach eliminates much overhead without affecting the final result.

LVMM utilizes a *pro-active* routing protocol, *Vehicular Multicast Routing Protocol* (VMRP), which maintains routes on a continuous basis. Thus, when a vehicle has to forward a multicast packet, the nodes to which it has to send the packet are already known and can be used immediately.
VMRP stores route information similar to routing protocols for static networks -- essentially, a routing table has an entry for each node N in a CTPG; each entry contains the nodes in the CTPG to which a vehicle has to forward packets received from N.
This way, the routing path followed by each multicast packet is a sequence of nodes that is not explicit: it corresponds to a next hop table lookup at each node along the route.

Upon receiving a packet each node knows if it has to forward the packet to other nodes and it knows the nodes to which it must forward the packet. No on-demand actions must be performed.
Having updated routing tables expedites packets forwarding and leaves space in packets for application layer data because the destinations of each packet must not be inserted in the packets' headers.
A pro-active state makes forwardings faster than no state keeping approaches and allows packet to carry more application data.

Summarizing, VMRP has the following features:

- it is *proactive*,
- utilizes only *local knowledge*, that is data in the CTPG of each node,
- it is *loop-less*,
- it utilizes the *least number of transmissions* to deliver a packet from each source to all the destinations and
- VMRP is scalable: *scalability* comes from the fact that it does not rely on any global topological information, and each node makes local forwarding decisions based solely on its neighbouring nodes (i.e. its CTPG).

It is *necessary and sufficient* condition that each node knows it own CTPG for a given MMG: global knowledge is not required.
LVMM and VMRP are also the subjects of [88].

8.5.1 Formalization of VMRP

A packet must be delivered to all members Within a MMG.
A packet contains no explicit receiver(s) but each node must forward it so that it is finally received by all members.
A node can send a packet only to directly reachable vehicles: external nodes don't act as router and they are not needed because of *Property 1* (section 6.1).
Let *MMG(P)* be the *Mobile Multicast Group* for an application P and let N be the number of its members.
As exposed in section 3 of this chapter, at a given interval of time t, the *MMG(P)* can be modelled by a *MMG-graph* $\mathbf{MMG_t} = (\mathbf{X},\mathbf{E})$, where $\mathbf{X} = \{\mathbf{x_1},\ldots,\mathbf{x_n}\}$ and $\mathbf{E} = \{(\mathbf{x_i},\mathbf{x_j}) : |\mathbf{x_i} - \mathbf{x_j}| < \mathbf{R},\ \mathbf{x_i} \in \mathbf{X},\ \mathbf{x_j} \in \mathbf{X},\ (\mathbf{i} \neq \mathbf{j})\}$.
Remember that $j\,(\mathbf{ID}) = \mathbf{x_i}$; let $j'(\mathbf{x_i}) = \mathbf{ID}$, that is, given the position of a vehicle, j' returns the identifier of the vehicle: j': *Position* $\rightarrow$ *Identifier*.
For each member of *MMG(P)*, let *CTPG* denotes the *Chosen Time-Proximity Group* associated to *MMG(P)*.

Let p be a multicast packet sent by a source vehicle that must be received by all other members. The following notation means that a vehicle with position $\mathbf{x_i}$ sends a multicast packet p; the vehicle $j'(\mathbf{x_i})$ is the source of the packet:

(1) *Source_sends* $\qquad j'(\mathbf{x_i}) \xrightarrow[\mathbf{s}]{} \mathbf{p}$

The following notation represents the fact that a multicast packet sent by a source node $\mathbf{x_i}$ is received by all the members of a MMG.

(2) *Result* $\qquad j'(\{\mathbf{X} - \mathbf{x_i}\}) \longleftarrow \mathbf{p}$

In other words, all vehicles unless one (the source) receive a multicast packet p.
Notation $j'(\mathbf{x_j}) \longleftarrow \mathbf{p}$ means that a vehicle with position $\mathbf{x_j}$ receives once a multicast packet p, i.e. with no duplicates; (2) applies that syntax to the set of recipients.

The result of the multicasting realized by VMRP is expressed by the above (1) and (2) and has the features listed in the previous paragraph.

How VMRP performs its job and how it achieves the aforementioned features is described below, for this purpose we introduce the following notation.

$j'(\mathbf{x_i}) \longrightarrow (\mathbf{p}, CTPG)$: the vehicle with position $\mathbf{x_i}$ sends the packet p to all vehicles in its *CTPG*.

$j'(\mathbf{x_i}) \longleftarrow (\mathbf{p}, \mathbf{x_j})$: the vehicle with position $\mathbf{x_i}$ receives the packet p from the vehicle with position $\mathbf{x_j}$.

$j'(\mathbf{x_i}) \underset{\neg}{\longrightarrow} (\mathbf{p}, \mathbf{x_j})$: the vehicle with position $\mathbf{x_i}$ does not forward the packet p to the vehicle with position $\mathbf{x_j}$.

$j'(\mathbf{x_i}) \underset{\mathbf{sx}}{\longrightarrow} \mathbf{p}$: the vehicle with position $\mathbf{x_i}$ forwards the packet p to all the nodes of its *CTPG* that are at its left, that is, behind it.

$j'(\mathbf{x_i}) \underset{\mathbf{dx}}{\longrightarrow} \mathbf{p}$: the vehicle with position $\mathbf{x_i}$ sends the packet p to all the nodes of its *CTPG* that are at its right, that is, in front of it.

Sending Packets

When a source of multicast traffic has packets to send it invokes the following operation for each packet p it sends: $j'(\mathbf{x_i}) \longrightarrow (\mathbf{p}, CTPG)$ (VMRP's *Source_sends*). Thus, with VMRP when a source of multicast traffic has a packet to send, it sends the packet to all the vehicles in its *CTPG* for the specified MMG. We have $j'(\mathbf{x_i}) \underset{\mathbf{s}}{\longrightarrow} \mathbf{p} = j'(\mathbf{x_i}) \longrightarrow (\mathbf{p}, CTPG)$.

Forwarding packets to achieve correctness and avoid duplicates with the minimum number of transmissions: the final result of VMRP

Each node follows rules (a) and (b) upon receiving a packet p:

(a) *if* $j'(\mathbf{x_i}) \longleftarrow (\mathbf{p}, \mathbf{x_j}) : \mathbf{j < i}$

then $j'(\mathbf{x_i}) \underset{\mathbf{sx}}{\longrightarrow} \mathbf{p} \Leftrightarrow EuclideanDistance(\mathbf{x_{i+1}}, \mathbf{x_j}) > \mathbf{R}$

(b) *if* $j'(\mathbf{x_i}) \longleftarrow (\mathbf{p}, \mathbf{x_j}) : \mathbf{j > i}$

then $j'(\mathbf{x_i}) \underset{\mathbf{dx}}{\longrightarrow} \mathbf{p} \Leftrightarrow EuclideanDistance(\mathbf{x_{i-1}}, \mathbf{x_j}) > \mathbf{R}$ •

Rule (*a*) establishes that when a node with position x_i receives a packet *p* from a vehicle $j'(x_j)$ in front of it, then, if and only if the very first vehicle behind it does not have a link with $j'(x_j)$, $j'(x_i)$ forwards *p* to *all* the vehicles in its *CTPG* that are *behind* itself, and only to those vehicles; otherwise $j'(x_i)$ performs no forwardings. Rule (*b*) establishes that when a node with position x_i receives a packet *p* from a vehicle $j'(x_j)$ behind it, then, if and only if the very first vehicle in front of it does not have a link with $j'(x_j)$, $j'(x_i)$ forwards *p* to *all* the vehicles in its *CTPG* that are *in front* of itself, and only to those vehicles; otherwise $j'(x_i)$ performs no forwardings.

8.5.2 Analysis of VMRP

In this section we analyze VMRP and we proof that it is correct, loop-free and delivers packets utilizing the minimum number of transmissions from each source to every other member. Next paragraph describes how VMRP implements this behaviour.
Remember that X is an ordered set, that is, all vehicles are ordered in a *MMG-graph*. Specifically, $x_j > x_h$ if $j < h$, that is, considering vehicles' identities, vehicle A "precedes" vehicle B where A = $j'(x_j)$ and B = $j'(x_h)$.
Nodes x_1 and x_n are *extreme* nodes, all other nodes are *intermediate* nodes.
The following lemmas are immediate from the properties of a *MMG-graph*.

Lemma 1 (Intermediate nodes). Because a *MMG-graph* is connected, an intermediate node can communicate with at least its very first left and right neighbours:

$$\forall x_i (i \neq 1 \wedge i \neq N \rightarrow \exists((x_i, x_{i+1}) \wedge (x_{i-1}, x_i)))$$

Lemma 2 (Connectivity).

if $\exists(x_i, x_{i+k}) : k > 1$

then $\forall w, v (i \leq w \leq i+k \wedge i \leq v \leq i+k \wedge w \neq v \rightarrow \exists(x_w, x_v))$ •

For example, if there exists a (bi-directional) link between a vehicle with position x_8 and a vehicle with position x_3, then, for Lemma 3, there exists a link between each pair of vehicles from x_8 to x_3 ($x_8, x_7, ..., x_3$; with i = 3 and k = 5).

Lemma 3 (Knowledge). If a vehicle $j'(x_i)$ receives a packet from a vehicle $j'(x_j)$ with $x_j > x_i$, then $j'(x_i)$ knows that all vehicles in its *CTPG* that are in front of it: $\{j'(x_n) : n < i \wedge j'(x_n) \in CTPG\}$, received the packet.
In the same way, if a vehicle $j'(x_i)$ receives a packet from a vehicle $j'(x_j)$ with $x_j < x_i$, then $j'(x_i)$ knows that all vehicles in its *CTPG* that are behind it: $\{j'(x_n) : n > i \wedge j'(x_n) \in CTPG\}$, received the packet.

Theorem 1 (Loop avoidance). If each node follows rules (*a*) and (*b*) upon receiving a packet, then the routing protocol is loop-less .
Proof. Rules (*a*) and (*b*) implicitly establish that each node, upon receiving a multicast packet *p*, never forwards that packet to vehicles that are at its right (i.e. in front of it) if it received *p* from a node at its right; and, each node does not forward *p* to vehicles that are at its left (i.e. behind it) if it received the packet from a node at its left; that is, the following two rules are respected:

$$(c) \quad \text{if } j'(x_i) \longleftarrow (p, x_j): j < i \text{ then } j'(x_i) \xrightarrow{\neg} (p, x_h): h < i$$

$$(d) \quad \text{if } j'(x_i) \longleftarrow (p, x_j): j > i \text{ then } j'(x_i) \xrightarrow{\neg} (p, x_h): h > i \quad .$$

A loop can exist if and only if a node sends back a packet to a node that already received it, that is, two cases are likely that generate a loop:

$$(1) \quad j'(x_i) \longleftarrow (p, x_j): j < i \wedge j'(x_i) \longrightarrow (p, x_h): j \le h < i$$

$$(2) \quad j'(x_i) \longleftarrow (p, x_j): j > i \wedge j'(x_i) \longrightarrow (p, x_h): i < h \le j$$

We have that rule (*c*) avoids case (1) and rule (*b*) avoids case (2).

Because of (*a*) and (*b*), packets sent by a member travel only along two directions: toward the leftmost extreme or toward the rightmost extreme.
Packets travelling toward the leftmost extreme are always sent by a vehicle to vehicles behind it; they form a *left flow* of packets. In a *left flow* each vehicle receives a packet

from a vehicle in front of it (at its right) and if necessary forwards that packet to vehicles behind it (at its left).
Packets travelling toward the rightmost extreme are always sent by a vehicle to vehicles in front of it; they form a *right flow* of packets. In a *right flow* each vehicle receives a packet from a vehicle behind it (at its left) and if necessary forwards that packet to vehicles in front of it (at its right).

Definition (right flow and left flow). For each vehicle $j'(\mathbf{x_i})$ a *right flow* ($flow_{dx}$) and a *left flow* ($flow_{sx}$) of packets are respectively the followings:

$$flow_{dx}(\mathbf{x_i}) = \left\{ \mathbf{p} : j'(\mathbf{x_i}) \longleftarrow\!\!\!\!- (\mathbf{p}, \mathbf{x_j}) \wedge \mathbf{j > i} \right\}$$

$$flow_{sx}(\mathbf{x_i}) = \left\{ \mathbf{p} : j'(\mathbf{x_i}) \longleftarrow\!\!\!\!- (\mathbf{p}, \mathbf{x_j}) \wedge \mathbf{j < i} \right\} \quad .$$

Lemma 4 (flows' generation). An intermediate node that is a source of multicast traffic generates both a *right* and a *left flow*. The extreme node $j'(\mathbf{x_N})$ generates only a *right flow*; the extreme node $j'(\mathbf{x_1})$ generates only a *left flow*.

Lemma 5. If a node $j'(\mathbf{x_i})$ receives a packet $\mathbf{p} \in flow_{dx}(\mathbf{x_i})$ then $j'(\mathbf{x_i})$ can only forward **p** to a node at its right, that is, in front of it. If a node $j'(\mathbf{x_i})$ receives a packet $\mathbf{p} \in flow_{sx}(\mathbf{x_i})$ then $j'(\mathbf{x_i})$ can only forward **p** to a node at its left, that is, behind it.

Theorem 2 (shortest path tree). If each node applies rules (*a*) and (*b*) then the final routing is correct, loop-free and utilizes the minimum number of transmissions to deliver packets from each source to every other member .

Given an *MMG-graph* $\mathbf{MMG_t} = (\mathbf{X}, \mathbf{E})$ and a source s, rules (*a*) and (*b*) determine a minimum-cost tree for **X** i.e., they determine a tree $\mathbf{G} = (\mathbf{X}, \mathbf{E'})$ rooted at the source, s, of multicast packets by utilizing a subset of arcs $\mathbf{E'} \subset \mathbf{E}$ which connects any two nodes of **X** with minimum number of transmissions. We assumed that all links have the same associated cost; this way, cost is expressed as the number of required transmissions.
G is the *shortest path tree* from a source to all other nodes: there is a single minimal-cost path between s and every other member, that is, rules (*a*) and (*b*) guarantees correctness and duplicates' avoidance utilizing the minimum number of transmissions.

Proof. From rules (*a*) and (*b*) we have that a *left* and *right flow* is generated for each packet. These two flows are able to deliver the packet to all nodes in the multicast group as immediately follows from Lemma 1 and Lemma 2: **G** reaches all members. We must proof that with rules (*a*) and (*b*) we utilize a shortest path to deliver packets from each source to every destination, that is $\mathbf{E}'$ is a set of unidirectional links and **G** is a valid solution to the *shortest path tree* problem: this way the protocol is correct, loop-free and utilizes the minimum number of transmissions.

Remember that cost is expressed as the number of required transmissions: all links have cost 1. Let $\mathbf{x_S}$ be the source node, let $\mathbf{d_{x_j}}$ be the cost of the (unique) path from $\mathbf{x_S}$ to a node $\mathbf{x_j}$ in **G**, i.e. $\mathbf{d_{x_j}}$ is the sum of the links that from the source reach $\mathbf{x_j}$ and let $\mathbf{d} = \mathbf{[d_j]}$ be the vector of the cost of the paths in the tree **G**; we have that $\mathbf{d_{x_s}} = 0$. Finally, let $\langle \mathbf{x_j}, \mathbf{x_h} \rangle$ denote the link that allows node with position $\mathbf{x_j}$ to send data to node with position $\mathbf{x_h}$, but not vice versa and let $\mathbf{c}\langle \mathbf{x_j}, \mathbf{x_h} \rangle$ be the cost of a unidirectional link; it is $\mathbf{c}\langle \mathbf{x_j}, \mathbf{x_h} \rangle = 1$.

G is a valid solution for the *shortest path tree* problem if and only if **d** verifies the following conditions of Bellman:

$$\mathbf{d_{x_j}} + \mathbf{c}\langle \mathbf{x_j}, \mathbf{x_h} \rangle \geq \mathbf{d_{x_h}}, \quad \forall \langle \mathbf{x_j}, \mathbf{x_h} \rangle \in \mathbf{E}.$$

Clearly, those conditions will be satisfied for equality for each unidirectional link $\langle \mathbf{x_j}, \mathbf{x_h} \rangle \in \mathbf{E}'$.

If **G** is a valid solution for the *shortest path tree* problem, then if it contains the link $\langle \mathbf{x_j}, \mathbf{x_h} \rangle$ it is $\mathbf{d_{x_j}} + \mathbf{c}\langle \mathbf{x_j}, \mathbf{x_h} \rangle = \mathbf{d_{x_h}}$; in fact the only path from $\mathbf{x_S}$ to $\mathbf{x_h}$ is made up of the path from $\mathbf{x_S}$ to $\mathbf{x_j}$, whose cost is $\mathbf{d_{x_j}}$, and of the link $\langle \mathbf{x_j}, \mathbf{x_h} \rangle$ whose cost is $\mathbf{c}\langle \mathbf{x_j}, \mathbf{x_h} \rangle = 1$.

We will see how rules (*a*) and (*b*) determine the links removal from the set **E** to $\mathbf{E}'$.

All starts with the *Source_sends* method. Let $\mathbf{x_m}$ be the left-most node of the *CTPG* of $\mathbf{x_S}$ and $\mathbf{x_n}$ be the right-most node of the *CTPG* of $\mathbf{x_S}$. A packet *p* is delivered from the source to all the nodes in its *CTPG* through links that cost 1, that is, one transmission (clearly this is the minimum possible cost). That is, $\mathbf{d_{x_j}} = \mathbf{d_{x_s}} + \mathbf{c}\langle \mathbf{x_S}, \mathbf{x_j} \rangle, \quad \forall \mathbf{x_j} : \mathbf{x_m} \leq \mathbf{x_j} \leq \mathbf{x_n} \wedge \mathbf{x_j} \neq \mathbf{x_S}$. *Suorce_sends* adheres to Bellman conditions, because rules (*a*) and (*b*) eliminate all links $\langle \mathbf{x_j}, \mathbf{x_S} \rangle$ (with (*c*) and

(*d*)) and they eliminate all links $\langle x_i, x_a \rangle : x_m < x_i < x_s \wedge x_a < x_i$ and all links $\langle x_i, x_a \rangle : x_s < x_i < x_n \wedge x_a > x_i$. Thus, each node in the *CTPG* of x_s receives once the packet.

At this point **G** utilizes only the links $E' = \{\langle x_s, x_j \rangle : x_m \leq x_j \leq x_n \wedge x_j \neq x_s\}$ and each node x_j that have received the packet has minimum d_{x_j}.

If the source has reached all nodes, then the multicasting of the packet is complete and **G** is a *shortest path tree*. Otherwise other steps are required:

(*) If some node must still receive the packet *p*, then only x_m or x_n (rules (a) and (*b*)) can forward the packet to those nodes: that is, only x_m and x_n can add a link to E'.

If some nodes $\{x_w : w > m\}$ and $\{x_v : v < n\}$ are member of the MMG, they must still receive the packet. x_m forwards the packet to the nodes in its *CTPG* behind it: $\{x_w : w > m \wedge x_w \in CTPG\}$; it then adds to E' the unidirectional links $\langle x_m, x_w \rangle$; because d_{x_m} is minimum, each $d_{x_w} = d_{x_m} + 1$ is minimum.

x_n forwards the packet to the nodes in its *CTPG* in front of it: $\{x_v : v < n \wedge x_v \in CTPG\}$; it then adds to E' the unidirectional links $\langle x_n, x_v \rangle$; because d_{x_n} is minimum, each $d_{x_v} = d_{x_n} + 1$ is minimum. Bellman conditions are still valid for **G**.

The steps at (*) are repeated until an extreme node is reached, where x_m is always the left-most reached node, while x_n is always the right-most reached node.

Bellman conditions are valid for each $\langle x_j, x_h \rangle \in E'$.

We have that rules (*a*) and (*b*) guarantee correctness and duplicates' avoidance utilizing the minimum number of transmissions. •

8.5.3 The proactive algorithm

VMRP follows rules (*a*) and (*b*) when a node receives a packet; the result is that VMRP is correct, loop-less and utilizes the minimum number of transmissions.
VMRP implements rules (*a*) and (*b*) with a proactive algorithm that builds, at each node, the routing table for a given MMG.
By maintaining a routing table, each node, upon receiving a packet, knows which forwardings it has to perform and no on-demand operations are needed; this way, packets' forwarding is fast: each node, upon receiving a packet does not make any computation, but simply indexes a routing table with the sender of the packet as the key and discovers which forwardings it has to perform.
Each node runs the same algorithm that pro-actively computes a routing table for each MMG to which the node have joined
The algorithms utilized by the multicast protocol are described below; they build an optimized graph for every source of packets by having each node just knowing its CTPG.
In the following a routing table is implemented with a hash-table, called *my_forwardings*; each table must be pro-actively updated, that is why, the core function of VMRP in each node is to keep updated each *my_forwardings* table. A routing table has the following form:

- **Hashtable < Identifier, List<Identifier>* > my_forwardings;**

  This table holds the state kept at each node that allows fast processing of packets. It is pro-actively maintained so that it is always updated and correctly reflects the actual state of the underlying network.
  Each node has its *my_forwarding* table for each MMG and only its neighbouring nodes are present in this instance: this is because only the node in the CTPG are needed to be monitored to achieve a global source-optimized multicasting.
  Each node runs the same algorithm that updates the table for a MMG.
  The key of this table is the Identifier of the sender node (the neighbour from which a node receives packets). Only the sender node is needed to decide how to route a packet.
  The value is a list of all neighbours to which this node must send a packet received by the node stored as the key.

Sending packets

When a vehicle has multicast packets to send to the members of a MMG(P), i.e. it is the source of multicast traffic of the application P, it utilizes the above described *Source_sends* (Lemma 1) and sends the packets to all the nodes contained in its CTPG relative to MMG(P). In the code below the MMG(P) is identified by the parameter *groupID*.

```
/* Method that a source of multicast packets invokes when it
has a packet to send.
Parameter 1 : the multicast packet to send.
Parameter 2 : the MMG to which the packet belongs. */
Source_sends( Packet packet, MMG_Identifier groupID ){

    /* groupID identifies MMG(P). CTPGs contains a CTPG for each
    multicast application that this node runs.
    The sending node has an updated CTPG for an application P, which
    contains the identities of its neighbours that are members of MMG(P);
    this method obtains a list of all the identifiers of the nodes contained in
    the CTPG of application P. */
    List neighbours = CTPGs[groupID].Identifiers();

    /* This node transmits the packet to all the nodes in its CTPG */
    Send( packet, neighbours );
}
```

The above method finds the list of all vehicles in the *Chosen Time-Proximity Group* for a given application P of the invoking node and sends a packet to those vehicles.
The *Send* operation allows a vehicle to transmit a packet to all nodes in its CTPG, through the MAC and Physical Layers.

Receiving packets

When a node receives a packet it performs two actions:

- it takes the packet for itself and dispatches it to the upper layer and
- it inspects a routing table to find out if it has to forward the packet to some of its neighbours.

This second operation is what allows multicasting among all members: each node not only keeps a packet for itself, it also acts as a router for other nodes in order to deliver a packet to all the destinations.

When a vehicle receives a packet it will invoke the following methods:

```
if(isPacketLegitim(Packet& packet)){

    receive(packet);        // it inspects the packet and dispatches it to upper layer

    /* The sender is obtained by looking at the Sender field inside the packet's
    header. The Sender field holds the neighbour that has sent the packet to this
    node, not the node that have generated the packet (Source).
    GroupID is the identifier of the multicast group; it allows nodes to know to
    which group (and application) the packet belongs.*/
    forward(packet.sender, packet.GroupID, packet);
}
```

The *forward* method is described below. It inspect the routing table of the MMG to which the packet belongs and pass the packet to the lower layers to physically send it to the recipient(s).
The routing table is the aforementioned *my_forwardings* data structure, whose keys are the senders of a packet and the value is a list containing all the receivers (neighbours to which a packet sent by the key-node must be forwarded).

```
void forward(Vehicle_Identifier neighbour_sender, MMG_Identifier groupID,
                    Packet& packet){

 /* A node can be member of more than one MMG.
    Tables holds a routing table for each MMG this node is a member of.
    groupID is the identifier of the MMG to which packet belongs.
    my_forwardings is the routing table for the MMG of packet. */
    Hashtable my_forwardings = Tables[groupID];

    /* Take the list of neighbours that must receive packet if the sender is
    neighbour_sender */
    List forwardings =  my_forwardings[neighbour_sender];

    if( forwardings != null )
```

```
            /* Send the packet to all the neighbours found in the list through the
            MAC and Physical Layers */
            send( packet, forwardings );
}
```

Upon receiving a packet, a node already knows the actions it has to perform: this can be done thanks to the following method that is continuously invoked to keep a routing table up-to-date.
The following method, *UpdateState*, here not written in a formal language, will be invoked upon receiving new updated values about the actual topology from the lower sub-layer; it computes the new multicast routing table of a given MMG.
Given the last updates about the members of a CTPG, this method determines the forwardings that a vehicle has to perform upon receiving a packet from a given neighbour. The following method adheres to rules (*a*) and (*b*) and thus the final result is an optimum source based routing: each node builds an optimus local graph for each one of its neighbours member of a given MMG. The state is valid because it is pro-actively updated. It allows fast packet processing because forwardings are based solely on the neighbour who sent a packet: for each received multicast packet, the only information a node has to control is the last sender of the packet, i.e. the neighbour who sent the packet to this node.
When this method returns, the routing table will contain an entry for every node in a CTPG. Each neighbouring node is the key of an entry and it represents a vehicle that can send packets to this node.
The value associated to every key A is a List of all vehicles to which this node must forward a packet that it will receive from A. Such a List may be empty.
Let's consider a node X that is in a CTPG of node Y and assume that node Y has to forward packets received from X.
We have two cases:

- X is behind Y: the packets coming from X are a *right flow* for Y.
  Y will forward packets coming from X to all neighbours that are in front of itself.
- X is in front of Y: the packets coming from X are a *left flow* for Y.
  Y will forward packets coming from X to all neighbours that are behind itself.

Thus, if a node has to forward packets from a node X then it will forward those packets to all nodes in front or behind itself, if those packets are part of a *right* or *left flow* respectively.
Otherwise it has to perform no forwardings.
The method *UpdateState* is the following:

```
/* Updates the routing table, i.e. the hash-table my_forwardings, for
```

```
the specified MMG groupID */
void UpdateState(MMG_Identifier groupID){

    /* Delete old state upon receiving new updated values. Deletes all old elements.
       Tables holds a routing table for each MMG this node is a member of.
       my_forwardings is the routing table to be updated */
       Hashtable my_forwardings = Tables[groupID];
       my_forwardings.clear() ;

       /* Acquire the CTPG associated to the given MMG */
       CTPG = CTPGs[groupID];

       /* Each node will physically receive only from a node in its CTPG and will have
       to directly forward a packet only to its neighbours. Thus in order to compute its
       routing table it only has to control the nodes in its CTPG to determine the
       actions it has to perform upon receiving a packet. Each node in the CTPG will
       have an entry in the hash-table my_forwardings. The correspondent value
       associated to an entry H will be the identities of the nodes to which this node
       forward a packet received from H. */
       ∀n ∈ CTPG {

            /* This means: "If the node n is behind me, with respect to roadway's
            direction".
            This is important because if this node receives packets from a node
            behind it, then this node knows that if it has to forward the packet, then
            all of the forwardings will be directed only in front of it. Therefore this
            node will check only neighbours in front of it. This means that a right
            flow comes from node n. */
            if( n.GPSposition < this.GPSposition ){

                 /* Acquires information about the very first vehicle at its right,
                 because we have a right flow of packets */
                 Vehicle nearest_RIGHT_neighbour =
                 CTPG.findNearestNode(RIGHT);

                 /* Implements rule (b).
                 Test if the very first node in front of this node receives packets
                 from n: this states if this node has to forward the packets coming
                 from n. If this vehicle has to perform some forwardings, then the
```

```
        node has to send the packets from n to all nodes in front of it,
        because this is a right flow of packets. */
        if (EuclideanDistance ( n, nearest_RIGHT_node.GPSposition ) ≥
        R){
                /* Take every neighbour farther away from n than this
                node. In the MMG-graph, node n is behind this node */
                ∀ m ∈ CTPG : m.GPSposit ion > this.GPSpo sition {

                        /* Remember that whenever this node receives a
                        packet from node n, it must forward the packet,
                        i.e. act as router, to node m */
                        my_forwardings[n].append[m];

                        }       // end ∀ m
                }       // end if( EuclideanDistance...)
        }       // end if( n.GPSposition < this.GPSposition )

/* This means: "If the node n is in front of me, with respect to roadway's
direction". This is important because if this node receives packets from a
node in front of it, then this node knows that if it has to perform some
forwardings, then all of them will be directed only behind it. Thus this
node will check only neighbours behind it. This means that a left flow
comes from node n. */
else {

/* Acquires information about the very first vehicle at its left. Because
we have a left flow of packets */
Vehicle nearest_LEFT_neighbour = CTPG.findNearestNode(LEFT);

        /* Implements rule (a).
        Test if the very first node behind this node receives packets from
        n: this states if a node should forward packets. If it has to perform
        some forwardings, then this node has to send a packet to all nodes
        behind it because this is a left flow. */
        if ( EuclideanDistance( n , nearest_LEFT_node.GPSposition ) ≥
        Radius){

                /* Take every neighbour farther away from n than this
                node. In the MMG-graph, node n is in front of this node.
                */
```

```
                                    ∀ m ∈ CTPG : m.GPSposit ion < this.GPSpo sition {

                                            /* Remember that whenever this node receives a
                                            packet from node n it must forward the packet, i.e.
                                            act as router, to node m */
                                            my_forwardings[n].append[m];

                                    }       // end  ∀ m
                            }       // end if( EuclideanDistance...)
                    }       // end else
            }       // end Update()
```

8.5.4 An example

The previous paragraphs explained that:

- Each node only has to know about its neighbours' (nodes in its CTPG) identities and positions.
- Each node considers only its neighbourhood and computes an optimum graph for each of its *neighbours*. These graphs form the routing table of the node.
- The overall result is that we achieve an optimum source-based multicasting; this is done at each node with only local information, without a global knowledge of all other non directly reachable members.
- Each time the mobility prediction algorithm states that an update must be performed, each node computes a new routing table for all of its neighbours, so that the global multicast state is always up-to-date.

Each node has an updated view of its neighbourhood and maintains an updated routing table thanks to the *UpdateState* method.
When a vehicle is the source of multicast traffic it invokes the *Source_sends* method to send its packets.
When a vehicle receives a multicast packet it follows the reception procedure that adheres to rules (*a*) and (*b*).
This section gives an example of the *pro-active* nature and the optimum results, in compliance with Theorem 2, of VMRP.

The below figure 7.9 represents a MMG made up of five members that run the same multicast application P. Figure 7. 8 represents the real scenario on the roadway from which the MMG is abstracted.

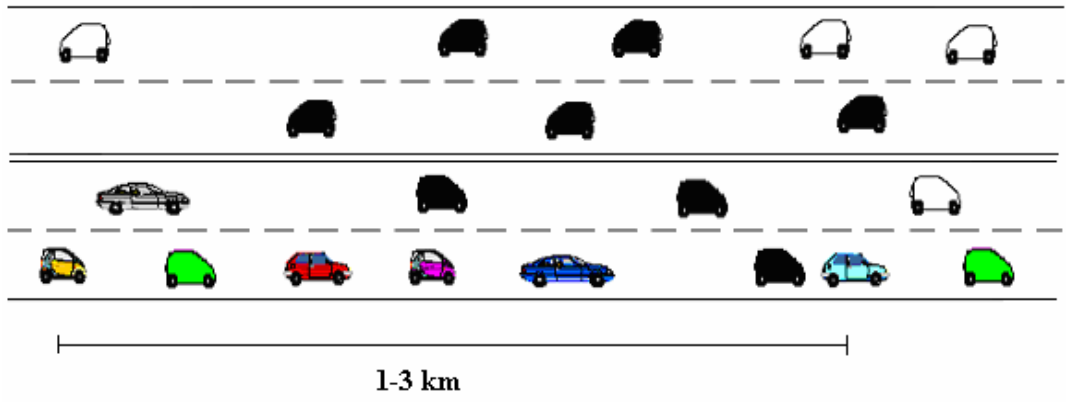


***Figure 7.8**. A real scenario of a stretch of roadway with a MMG and other non member vehicles. White, black and green cars represent vehicles that do not participate to the MMG. At the MMG sub-layer the underlying topology is abstracted taking into account only MMG's members*

The following figure shows a *Mobile Multicast Group* made up of six vehicles. We take into consideration a straight stretch of roadway, because the environment can be a highway and because, as long as radio transmissions are not disturbed, the underlying road topology is of no matter, as previously shown.

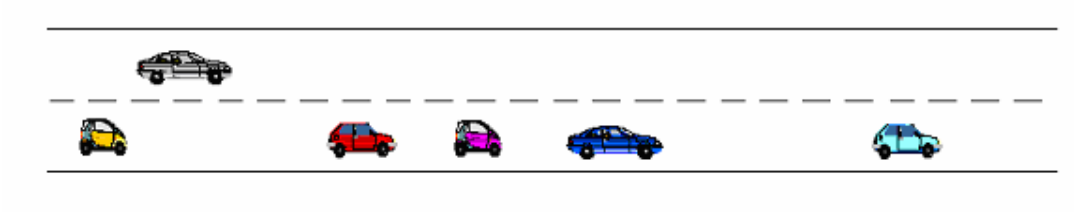


***Figure 7.9** . A MMG along a roadway. Six vehicle have joined this MMG.*

The following figure shows vehicles' transmission ranges. Transmission ranges cover all roadway's width and have a radius that allows the vehicles to reach nodes behind and in front of them.

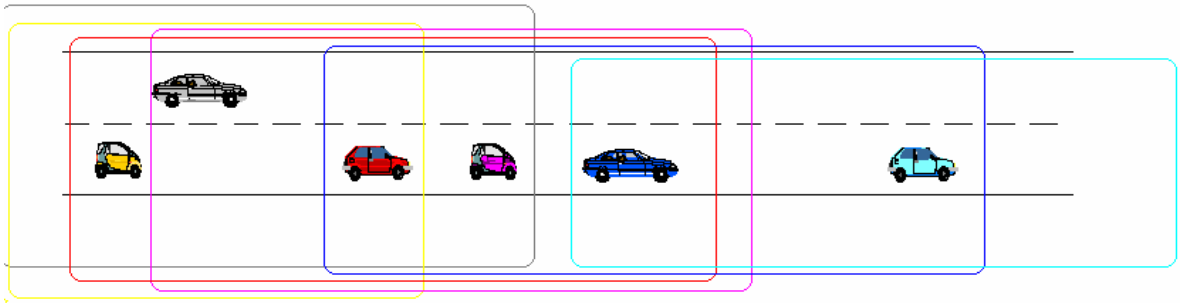


***Figure 7.10**. Transmission ranges of all members of a MMG.*

The following figure shows the wireless links that exist between all members of the example. Thick lines represent bidirectional links between two directly reachable vehicles. Each node has all directly reachable vehicles in its CTPG.

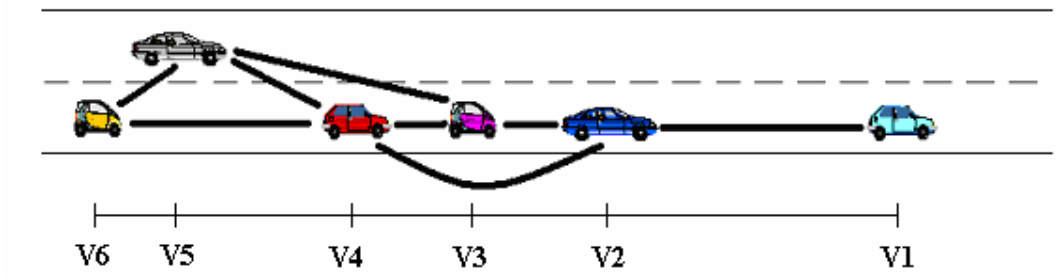


***Figure 7.11**. Wireless links between members of the MMG.*

Because of radio transmission ranges and GPS devices, we can project vehicles on a straight line where they are ordered through their coordinates.
Therefore we can consider vehicles as all lying on a straight line on which they are distinguished by their identities and positions (i.e. coordinates given by GPS devices – or other position tracking devices).

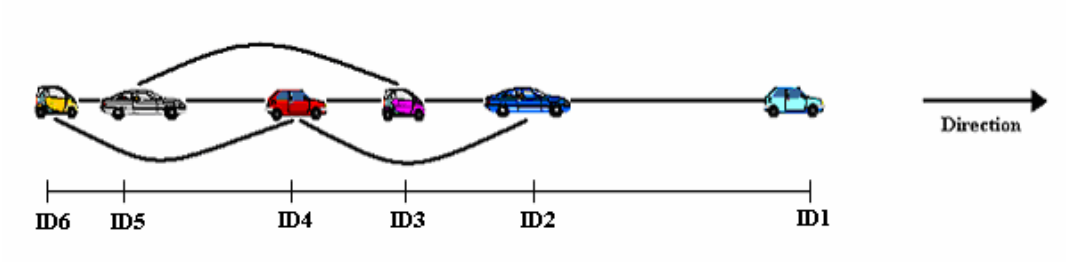


***Figure 7.12**. Members of the MMG can be classified on a straight line through their coordinates and each node knows its directly reachable neighbours.*

Each vehicle has the following *Chosen Time-Proximity Group* for the group of the example :

- CTPG(V6, τ, α, MMG(P)) = {V5, V4}
- CTPG(V5, τ, α, MMG(P)) = {V6, V4, V3}
- CTPG(V4, τ, α, MMG(P)) = {V6, V5, V3, V2}
- CTPG(V3, τ, α, MMG(P)) = {V5, V4, V2}
- CTPG(V2, τ, α, MMG(P)) = {V4, V3, V1}
- CTPG(V1, τ, α, MMG(P)) = {V2}

Let's consider the following examples of optimized trees:

- If the yellow car (ID6) was a source of multicast traffic, then the path followed by the packets originated at vehicle ID6 would follow the source-optimized graph depicted in figure 7.13.

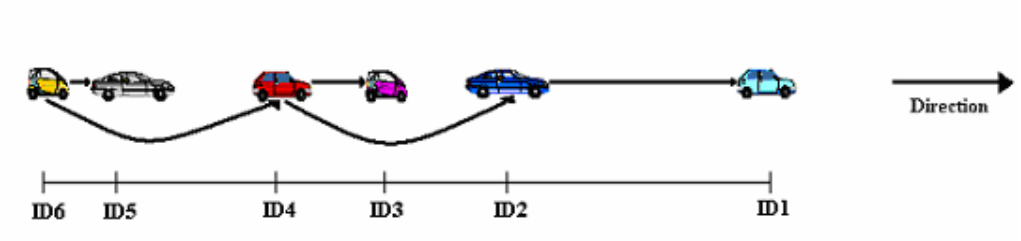


***Figure 7.13**. Path followed by packets originated at vehicle ID6.*

The above tree is of minimum cost because it requires the minimum number of transmissions in order to deliver a packet sent from the source to all other members, it is avoids loops and duplicates (Theorem 2).
In order to multicast a packet to all vehicles, three transmissions are required: the first transmission delivers the packet to grey (ID5) and red (ID4) vehicles,

they are both in yellow car's transmission range; then red car sends the packet to purple (ID3) and blue (ID2) vehicles; with the third transmission, the blue node sends the packet to the turquoise node (ID1).

- If the red car (ID6) was a source of multicast traffic, then the path followed by the packets originated at vehicle ID46 would follow the source-optimized graph depicted in figure 7.14.

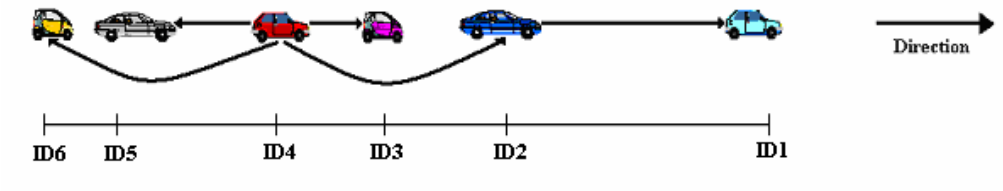


Figure 7.14*. Path followed by packets originated at vehicle ID6.*

Only two transmissions are required to deliver a packet to all members and we see that this is the best achievable result.

Each node follows the above described proactive multicast routing protocol VMRP, thus for the proposed example, in the specified interval of time, each vehicle will have the following routing tables for the MMG:

ID6's routing table

KEY	VALUE
ID5	
ID4	

We can see that yellow node (ID6) receives packets only from the grey (ID5) and red (ID4) vehicles, that is, it has ID5 and ID4 in its CTPG.
Because ID6 is an extreme of the *MMG-graph* it has to perform no forwardings upon receiving a multicast packet.
If it is the source of multicast traffic it executes the *Source_sends* method to send a packet and sends it to both ID5 and ID4.

ID5's routing table

KEY	VALUE
ID6	
ID4	
ID3	ID6

Grey vehicle (ID5) forwards a packet only if it comes from the purple (ID3) vehicle.

ID4's routing table

KEY	VALUE
ID6	ID3 ID2
ID5	
ID3	
ID2	ID5 ID6

ID4 has four vehicles in its CTPG. It will only forward packets that it will receive from ID6 and ID2.
If it is the source of multicast traffic it will send packets to ID6, ID5, ID3 and ID2.
The above routing table shows that, for the above example, ID4 has to forward packets that it receives from ID6 to ID3 and ID2, while it has to forward to ID5 and ID6 the packets received from ID2. It will not send any packet received from ID5 and ID3: this is because other nodes will deliver packets received from those vehicles in order to achieve a minimum cost multicasting.
We see that there is no entry for ID1; that is because that vehicle is not in red car's CPTG: if ID1 is the source of multicast packets or it forwards any packets toward red car, then for the red car those packets are a *left flow* of traffic as there is at least one vehicle that is between red car and turquoise car that links together those vehicles.
We can find in ID4 routing table the red node's actions that are depicted in figure 7.13.

ID3's routing table

KEY	VALUE
ID5	ID2
ID4	
ID2	

ID2's routing table

KEY	VALUE
ID1	ID3 ID4
ID3	ID1
ID4	ID1

Purple vehicle (ID3) will forward packets coming from ID5. It will perform no other forwardings as long as the topology remains the same.
Blue node (ID2) has ID1, ID3 and ID4 in its CTPG. It is the "anchor" for turquoise (ID1) node to the group: it is the only node inside the MMG that can communicate with ID1.

ID1's routing table

KEY	VALUE
ID2	

ID1 will not perform any forwarding upon receiving a multicast packet. It only has ID2 in its CTPG and thus if it is the source of multicast packets it will send its packets only to the ID2.

Let's take the case where ID6 is a source of multicast traffic. Let *p* be one of the multicast packets that must be delivered from ID6 to all other members.
The following actions describe how routing tables in each node achieve a global optimus multicasting.

1. ID6 executes the *Source_send* method for *p* and thus forwards the packet directly to all nodes in its CTPG.
   ID5 and ID4 receive *p*.
2. ID5 indexes its *my_forwardings* table with the source of the received packet and discovers that it has to perform no forwardings.
   ID4, instead, immediately forwards the packet to ID3 and ID2. This is done by the *forward (ID6, MMG-ID, p)* method that indexes the Hashtable *my_forwardings* with the sender of the packet (*my_forwardings[ID6]*) and transmits *p* to the set of nodes contained in the List.

3. Then, ID3 detects that it has to perform no forwardings upon receiving a packet from ID4, while ID2 forwards the packet *p* to ID1.
4. ID1 just receives the packet for itself and does not perform any forwarding.

All nodes receive the multicast packet and no node receives the same packet more than once. The final result is a minimum cost source-based multicasting. The route found is the same route depicted in figure 7.13 and it is in compliance with rules (*a*) and (*b*).

8.6 Global Membership knowledge

By utilizing VMRP, all vehicles maintain an updated view of the global membership, so that each vehicle knows all other members of a group.
The view of the global membership is maintained at each vehicle with a "soft state" in a decentralized manner: each node, independently from the others, computes the same global state.
Every vehicle N has a table (*Global_Membership_table*) containing every other member of a given group. A timeout is associated to each member vehicle: if it is not refreshed, that vehicle is no more considered a member of the group by node N.
Each vehicle periodically multicasts a membership message (*Refresh Membership*) and all nodes upon receiving it, update the relative table.
LVMM sends *Refresh Membership* messages transparently to applications.
The global view of a group is utilized for security purposes and it is also made available to applications.
From an application perspective, it is often desirable to know all the members of a group and to maintain a consistent view of the global membership across all the hosts in the group.
From [79] we learn the followings:

- Any situation that demands (for legal or technical reasons) the presence of two or more specific entities to carry out a task may impose the need for a consistent membership view.
- It is of utmost importance that (unlike IP multicast groups, for example), *group membership be known* at all times, even in unreliable or unsteady communication environments such as dynamic ad-hoc networks.
- Non-members should not be able to participate in group communication, requiring the concept of closed and determinate groups along with data encryption.

LVMM transparently maintains a global membership that is available to applications.
Besides encryption that can be utilized both at this layer and the application layer, LVMM supplies applications with the above mechanisms and properties: each node transparently has a global view of the group, this view is consistent among all members, non-members do not participate in group communication because a CTPG only contains directly reachable nodes and because of *Core Property 1*.
Messages coming from a node that is not in the *Global_Membership_table* will be discarded by the members of a group. *Global_Membership_table* is then also utilized to check received messages: each node has its own table and it performs the controls

utilizing its own instance; this way security checks are distributed and repeated so that injecting of “fake” nodes is very difficult.

8.7 Multicast packets’ format

This paragraph suggests a header for multicast packets sent and received with VMRP. The first four fields are necessary fields for the multicast routing protocol, the others can be utilized for QoS issues.

Multicast packet header

Type	Source	Sender	Group ID	Timestamp	Sequence number	Checksum

Figure 7.15*. Format of the MMG sub-layer header.*

Seven relevant fields of a multicast packet have been depicted above; here are their meanings:

- *Type* – this field holds the type of the multicast packet, e.g. the packet contains application level data or it is a *Refresh Membership* packet and contains network level data.
- *Source* – the originator of the multicast packet: the node that has created and has sent the packet into the group. Unlike Internet multicasting receivers know which the source of multicast traffic is. This field can also be used for authentication purposes.
- *Sender* – the neighbouring node from which this node has received the packet.
- *GroupID* – the identifier of the *Mobile Multicast Group* (MMG) to which the packet belongs. This allows the nodes to know to which application the packet belongs.
- *Timestamp* – derived from a sampling clock at the sender. It may be useful to remove packet jitter introduced in the network and to provide synchronous playout at the receiver.
- *Sequence number* – incremented by the source at each packet sent; it can be used to detect packet loss and to restore packet sequence.

- *Checksum* – utilized to verify the packet.

Receivers are not in the header: they may be many, too many. This way the header has a fixed size. A node knows that a neighbour has already received a packet by just looking at its CTPG and the transmission radius R: no duplicated packets are sent by the multicast protocol.

9. MMG's extension

LVMM can utilize a second approach along with the above described approach.
The second approach allows multicast communications among vehicles that are not linked by a set of intermediate nodes that execute the same multicast application: nodes that do not have neighbours that want to execute a multicast application can still execute that application with other vehicles through a stable chain of intermediate nodes; intermediate vehicles not interested in multicast traffic help other vehicles to exchange multicast packets.
This way, LVMM can link together vehicles that have similar mobility, but are not directly reachable and have no intermediate vehicles that execute the same multicast application. This approach also helps coping with partitions of a MMG by utilizing other stable nodes to keep a group connected.
In particular, two nodes A and B can utilize LVMM if there exists a sequence of stable nodes between A and B:nodes that are respectively one in the TPG of the other from A to B and vice versa and thus form a stable chain of vehicles. Such a chain

- involves nodes not running the multicast application of A and B,
- but just involves vehicles with similar mobility and thus
- allows a stable connection between A and B.

In the remainder of this section I will call "*router nodes*" the *stable vehicles* that compose a chain.
Router nodes

- are member of a given MMG, but
- they are not interested in multicast packets exchanged between members: they only provide the connectivity and act as routers for other vehicles of the MMG, which run a given multicast application.

Thus a MMG can contain vehicles that do not execute a multicast application but use LVMM to support the communication between other vehicles that run the application.
With this extension, a MMG is still a group composed by stable nodes and thus all the aforementioned properties of a group and the routing protocol are valid.
Only vehicles that are inside a given TPG for an application can join the MMG for the application; all nodes member of a MMG must have a member of the MMG in their correspondent CTPG; but such a node may be a *router node* that only links together distant vehicles and guarantees stability and availability of communications.
This second approach wants to create a sort of connection-oriented topology that involves some non-multicast nodes in order to enable low-latency multicast communications. A stable connection, guarantees that, at a minimum, a physical

connection exists between nodes, otherwise we only had a ”best-effort” physical topology and no stable or deterministic multicasting could be performed.
This approach requires more complexity in handling multicast groups, requires some nodes, not interested in receiving packets of a MMG, to keep state for group maintenance and to act as routers for multicast packets.
This approach cannot be applied if stable nodes that are not interested in multicasting do not want to act as routers for other nodes or a stable chain of vehicles cannot be found.
What is worth mentioning is that multicast packets will mostly be part of real-time communications, so that kind of traffic will not be a slightly load traffic and, if not forced, other nodes may not want to deal with that traffic.
The figure below shows a MMG containing *router nodes* beside other members.

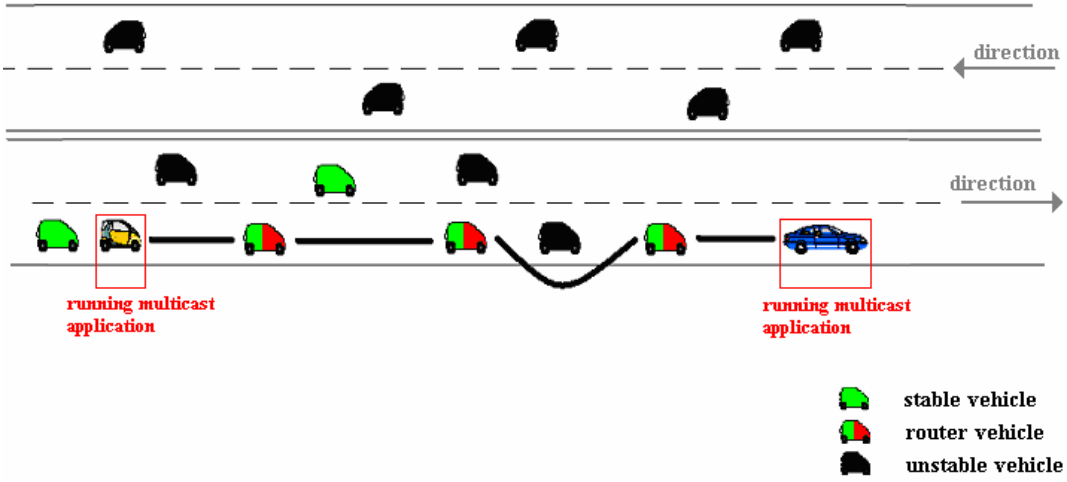


Figure 7.16*. LVMM enables multicast communication between vehicles that are not connected by a sequence of nodes that are interested in multicast traffic.*
Vehicles linked by a dashed line are members of the same MMG. Only the yellow and blue vehicle run a multicast application and are interested in multicast packets. Green vehicles are stable vehicles that are in the TPGs of other vehicles but are not member of the MMG. Green/red vehicles are green vehicles that are MMG' members not interested in multicast traffic: they are "router nodes".

The above figure shows a MMG made up of five vehicles. Only two vehicles, the yellow one and the blue one, run a multicast application and thus need multicast packets.
Red/green vehicles are member of the same MMG, but they are *router nodes* and thus allow the communication between the vehicles that run the multicast application.

With this second approach TPG membership could be relaxed to allow distant vehicles to communicate if the forwarding vehicle is slower than the back vehicle and they are distant enough to maintain a required communication before the rear vehicle overwhelms the leading node and the gap between the two nodes become too large. A stable chain must either way exist.
The above example shows that this second approach is much like the above described approach where intermediate members among two nodes are removed and substituted with *router nodes*. This way, it is like group membership is lowered by one level: not just vehicles that run a multicast application participate to a group, but other stable vehicle must participate to that group and act as routers.
A group is then identified by nodes that want to exchange multicast packets and also by other nodes that just act as routers.
"*Router nodes*" run the same multicast protocol as the other vehicles; the difference is that they only forward multicast packets.

10. LVMM vs. other multicast protocols

ODMRP [43, 52] and CAMP [52] are two multicast protocols designed for wireless mobile ad hoc networks; they were described in chapter 6.
They are recognized as some of the best multicast routing protocols for mobile ad hoc networks.
In this paragraph the differences between them and LVMM are shown, pointing out why they are not suitable for a low-latency multicasting in a vehicular network.

ODMRP and CAMP are mesh-based multicast routing protocols for mobile ad hoc networks and they are recognized as high robust against node mobility. Here are some of their common characteristics.

1. They are on demand protocols.
2. They try to solve the "Internet-like" multicast routing in an ad hoc network.
3. They are not specifically designed for a particular environment; then they do not take into account the peculiarities of vehicular networks.

ODMRP and CAMP utilize a mesh structure because, as authors rightly state, tree structures must be readjusted as connectivity changes and they require a global routing substructure such as link state or distance vector, which leads excessive channel and processing overhead.
On the other hand by exploiting the unique characteristics of a VANET, LVMM builds source-based multicast trees without the need of a global routing infrastructure. Each node only needs to know about its CTPG and the final result is a least-cost source-based multicasting.

ODMRP utilizes on demand routing techniques. Group membership and multicast routes are established and updated by the source "on demand". It has a request phase and a reply phase. While a multicast source has packets to send it floods a member advertising packet with data payload piggybacked. This packet (JOIN QUERY) is periodically broadcasted to the entire network to refresh the membership information and update the routes.
ODMRP suffers from excessive flooding when there are a large number of senders.

ODMRP and CAMP attempt to define the Internet multicasting model in an ad hoc environment. Their application is not bounded to a limited surface or area: they try to solve the "Internet like" multicast routing in an ad hoc network. They do not define a localized multicasting but a multicast that can spread boundless and then does require lots of non-member nodes to route messages.

In a VANET, ODMRP and CAMP would define a multicast protocol that is not constrained by geographic limits and clearly will not provide applications with any QoS because they do not matter about the underlying network. They try to connect very distant vehicles through the aid of other vehicles, but this approach does not guarantee QoS and if there is disconnection there is no communication at all.
A group is "weak" and "open" like it is in Internet-multicasting.
Because ODMRP and CAMP are not specifically designed for a given environment, they do not exploit the characteristic of a specific scenario such as VANETs.
They do not take into account nodes' mobility to find a stable topology nor they consider the underlying physical topology, availability or QoS.

A multicast routing protocol belonging to a different class of protocols is Position Based Multicast (PBM) [26]. It is a proposed solution for a multicast routing based on geographic routing.
PBM attempts to solve the same problem faced by ODMRP and CAMP: it provides a solution to the Internet-like multicasting for a wireless mobile ad hoc network.
The idea of PBM is to implement multicast routing exploiting position based routing to avoid both flooding and state maintenance.
With this protocol each node must know the positions of all of the destinations; these positions must be included in the packet or must be available locally by querying a Location Service, which in this case must be an all-to-all service, like DREAM's Location Service.
This assumption means that a multicast source inserts in each multicast packet all the receivers' identities, that it must know all the nodes that have joined a given group and also, if not available locally, it must insert in each packet, along with each node's address, each node's position.
PBM does not address problems like scalable distribution of group membership and position information.
It is not scalable, it was designed for an unbounded multicasting (Internet-like multicasting); it does not deal with the underlying network's topology; it does not take into account neither availability nor QoS.

On the other hand, LVMM defines an architecture and a routing protocol that exploits the characteristics of a vehicular network from the lower layers to the routing protocol layer. It is specific of a VANET and thus designed to exploit its unique characteristics. It takes into account the mobility model and builds upon it a routing protocol specific for a roadway scenario.
LVMM utilizes a pro-active state to keep updated its view of the underlying network because LVMM cares about low-latency and QoS. The routing protocol VMRP is specifically designed for a VANET and thus exploits its unique characteristics.

VMRP works in a bounded mobile area and allows only stable nodes to join a multicast group to grant lasting and low-latency communications (suitable for group cruise control, instant messaging, games, ...).

11. LVMM vs. Mobicast

The term *mobicast* [81] identifies a spatio-temporal variant of multicast.
The spatio-temporal character of mobicast relates to the obligation to deliver a message to all the nodes that will be present at time *t* in some geographic zone *Z*, where both the location and shape of the delivery zone are a function of time over some interval (*tstart, tend*).
Mobicast is a sort of *mobile Geocast*: all the nodes inside a given destination area must receive a packet; the destination area changes with time.
The focus of *mobicast* is not on the identities of the nodes but on the coordinates and the shape of the destination area.
There is no traditional notion of a 'group' because *mobicast* does not take into account the identities of the hosts.
A node is member of a group, in an interval of time *t*, if at time *t* it is inside the destination area. A host that resides within the destination area automatically becomes a member of the corresponding geocast group for the interval of time that it is inside the destination area and until the area does not change and the host no more resides within it.
Mobicast is a spatio-temporal mechanism because it is about a geographic area that changes its coordinates and shape along with time.
LVMM instead has its focus on the identities of the nodes.
LVMM is a spatio-temporal variable approach, too, but with a different meaning from the one associated to *mobicast*: with LVMM, the area covered by the mobile member nodes changes along with time; the communications always involve a group of relative stable nodes that moves fast and is never located within the same area in different times.
LVMM is something more than *mobicast* because it identifies and maintains MMGs that are

- spatio-temporal variable: the geographic area changes with time because a MMG is a dynamic group whose members continuously change their geographic positions;
- semi-stable about nodes' identities: a MMG is based on the identities of the mobile nodes; the identities of the recipients remain the same as long as the nodes do not leave the group (explicitly or implicitly due to their mobility).

Thus, LVMM handles groups of nodes where the word 'group' has the classical meaning of a set based on the identities of the nodes, but in a highly mobile environment.
VANETS are highly mobile networks but MMGs are semi-stable groups of nodes.

LVMM extracts from the underlying network a stable groups of nodes, thus it determines a Multicasting and not a Geocasting in the network. In order to perform its tasks, it utilizes the positions of the nodes, but its focus is on the identities.
Mobicast determines a Geocast inside the mobile destination area without a distinction among nodes inside the Geocast area.
LVMM and *mobicast* are both scalable because

- they do not rely on any global topological information, and
- each node makes local forwarding decisions based on its *spatial neighbourhood* configuration.

Conclusion

The subject of this thesis is multicasting in VANETs.
Vehicular networks will have great development and importance in the near future and multicasting will represent a fundamental aspect of communications in such networks.
This thesis is the result of the research and studies about vehicular networks and it describes a new solution to multicasting in VANETs that is not exposed in actual published documents.
The proposed solution is twofold.
First, a novel concept of multicasting has been applied to vehicular networks in order to guarantee a given degree of *availability* and stability to communications; otherwise no applications with QoS issues could be run in a highly mobile ad hoc environment.
Second, a middleware has been introduced and described that addresses all the tasks of multicast communications and deals with the underlying mobile network. The solution comprises a framework, LVMM, with distinct levels that address the specific problem of multicast communications in a vehicular ad hoc network and exploit the unique characteristics of such environment and of its nodes.
LVMM is "application-friendly" because it masks all the underlying complexity to the upper layers and handles all aspects of network communication from network's mobility handling to groups' creation and maintenance and routing protocol implementation and execution. In particular LVMM has a multicast routing protocol, VMRP, specifically designed for VANETs that is based upon a model that considers vehicles and roadways (*MMG-graph*). The proposed routing protocol is scalable and allows *low-latency* multicast communications.
The ultimate goal of LVMM is to assist software developers in their efforts to design and build multicast applications over vehicular ad-hoc networks.
This kind of multicasting allows the execution of many-to-many applications that otherwise would not have reason to exist in VANETs, because of the lack of stable communicating partners.
The key to LVMM strategy is to provide, at the application level, the appearance of stability in a domain that is characterized by high degrees of mobility and to provide applications with the mechanisms to transparently join/create groups and efficiently exchange multicast packets.

References

[1] Jun-Zhao Sun, "Mobile Ad Hoc Networking: An Essential Technology for Pervasive Computing", *Proc. International Conferences on Info-tech & Info-net, Beijing, China, C:316 – 321*, 2001.

[2] David B. Johnson, "Routing in Ad Hoc Networks of Mobile Hosts", *Proc. of 1st IEEE Workshop on Mobile Computing Systems and Applications (WMCSA'94)*, pp. 158--163, 1994.

[3] Petteri Kuosmanen, "Classification of Ad Hoc Routing Protocols", Finnish Defence Forces, Naval Academy.

[4] L.M. Feeney "A Taxonomy for Routing Protocols in Mobile Ad Hoc Networks", *in International Symposium on Handheld and Ubiquitous Computing (HUC'99), volume 8*, Karlsruhe, Germany, September 1999. ACM.

[5] RFC 2501, "Mobile Ad hoc Networking (MANET): Routing Protocol Performance Issues and Evaluation Considerations." S. Corson-University of Maryland, J. Macker - Naval Research Laboratory, 1999.

[6] David B. Johnson, David A. Maltz, "Dynamic Source Routing in Ad Hoc Wireless Networks", *in Mobile Computing (ed. T. Imielinski and H. Korth), Kluwer Academic*. Kluwer Academic Publishers, 1996.

[7] M. Scott Corson, Joseph P. Macker, Gregory H. Cirincione, "Internet Based Mobile Ad Hoc Networking", *IEEE Internet Computing, July-August 1999*, pp. 63-70.

[8] G. Finn, "Routing and Addressing Problems in Large Metropolitan-scale Internetworks", *Technical Report ISI Research Report ISU/RR-87-180, Inst. for Scientific Information*, Mar, 1987.

[9] Deering, S., and R. Hinden, "Internet Protocol, Version 6, Specification", RFC 1883, Xerox PARC, December 1995.

[10] Deering, S., and Hinden, R. "IP Version 6 Addressing Architecture", RFC 1884, Ipsilon Networks, Xerox PARC, December 1995.

[11] Tomasz Imielinski, Julio C. Navas, "Geographic Addressing, Routing, and Resource Discovery with the Global Positioning System", *in ACM MOBICOM*, Budapest, Hungary, September 26-30 1997.

[12] T. Imielinski and J. Navas, "GPS-Based Addressing and Routing", RFC 2009, Computer Science, Rutgers University, March 1996.

[13] Andersson, Christoffer, "GPRS and 3G wireless applications: professional developer's guide" *Published by John Wiley & Sons*, Inc. 2001.

[14] "How GPS receivers work", http://electronics.howstuffworks.com/gps.htm.

[15] M. Mauve, J. Widmer, "A Survey on Position-Based Routing in Mobile Ad Hoc Networks", *IEEE Network Magazine, 15(6):30--39*, November 2001.

[16] Silvia Giordano, Ivan Stojmenovic, Ljubica Blazevic "Position Based Routing Algorithms for Ad Hoc Networks: A Taxonomy", July 2001, http://www.site.uottawa.ca/ivan/routing-survey.pdf.

[17] P. Bose, P. Morin, I. Stojmenovic, J. Urrutia "Routing with Guaranteed Delivery in Ad Hoc Wireless Networks", *International Workshop on Discrete Algorithms and Methods for Mobile Computing and Communications (DIAL-M)*, pages 48–55, 1999.

[18] S. Datta, I. Stojmenovic, Jie Wu "Internal Node and Shortcut Based Routing with Guaranteed Delivery in Wireless Networks", *in Cluster Computing 5, pages 169--178*. Kluwer Academic Publishers, 2002.

[19] L. Barriere, P. Fraigniaud, L. Narayanan "Robust Position-Based Routing in Wireless Ad Hoc Networks with Unstable Transmission Ranges", *in Proceedings of the 5th international workshop on Discrete algorithms and methods for mobile computing and communications, pp. 19—27*, 2001.

[20] B. Karp, H. T. Kung "GPSR: Greedy Perimeter Stateless Routing for Wireless Networks", *in Proceedings of the Sixth annual ACM/IEEE International Conference on Mobile Computing and Networking (MobiCom 2000)*.

[21] Fabian Kuhn, Roger Wattenhofer, Yan Zhang, Aaron Zollinger "Geometric AdHoc Routing: Of Theory and Practice", *in Proc. Dial-M'02*, Atlanta, Georgia, USA, September 2002.

[22] S.Basagni, I. Chlamtac, V. R. Syrotiuk, B. A. Woodward "A Distance Routing Effect Algorithm for Mobility (DREAM)", *Proc. of ACM MobiComm'98*, October, 1998.

[23] H. Füßler, J. Widmer, M. Mauve, H. Hartenstein "A Novel Forwarding Paradigm for Position-Based Routing (with Implicit Addressing)", *in Proceedings of IEEE 18th Annual Workshop on Computer Communications (CCW 2003)*.

[24] Julio C. Navas, T. Imielinski "GeoCast - Geographic Addressing and Routing", *in ACM MOBICOM*, Budapest, Hungary, September 26-30 1997.

[25] K. Obraczka, G. Tsudilìk "Multicast Routing issues in Ad Hoc Networks", *IEEE International Conference on Universal Personal Communication, (ICUPC '98)*, October 1998.

[26] M. Mauve, H. Füßler, J. Widmer, T. Lang "Position-Based Multicast Routing for Mobile Ad-Hoc Networks", *Technical Report TR-03--004, Department of Computer Science, University of Mannheim*, 2003.

[27] Xia Jiang, Tracy Camp "A Review of Geocasting Protocols for a Mobile Ad Hoc Network", *Proceedings of the Grace Hopper Celebration (GHC '02)*, 2002.

[28] C. MaihÖfer "A Survey Of GeoCast Routing Protocols", *DaimlerChrysler AG, Research & Technology (RIC/TC) IEEE publications Second Quarter 2004, Volume 6, no. 2* .

[29] Y.B. Ko, N.H. Vaidya "Flooding-based Geocasting Protocols for Mobile Ad Hoc Networks", *in Proc. WMCSA New Orleans*, LA, 1999, pp. 471-480.

[30] Y.B. Ko, N.H. Vaidya "Geocasting in Mobile Ad Hoc Networks: Location-Based Multicast Algorithms", *in IEEE Workshop on Mobile Computing Systems and Applications (WMCSA'99)*, Feb. 1999.

[31] Wen-Hwa Liao, Yu-Chee Tseng, Kuo-Lun Lo, Jang-Ping Sheu "GeoGRID: A Geocasting Protocol for Mobile Ad Hoc Networks Based on GRID", *J. Internet Tech., vol. 1*, pp. 23-32, 2000.

[32] K. Seada, A. Helmy "Efficient Geocasting with Perfect Delivery in Wireless

Networks", *in Proc. IEEE Wireless Communications and Networking Conference (WCNC)*, Atlanta, Georgia, March 2004.

[33] I. Stojmenovic , M. Seddigh, J. Zunic "Dominating Sets and Neighbor Elimination-Based Broadcasting Algorithms in Wireless Networks", *IEEE Transactions on Parallel and Distributed Systems, vol. 12, no. 12* , December 2001.

[34] Min-Te Sun, W. Feng, Ten-Hwang Lai "Location Aided Broadcast in Wireless Ad Hoc Networks", *GLOBECOM 2001 - IEEE Global Telecommunications Conference*, no. 1, Nov 2001 pp. 2842-2846.

[35] C. Maihofer, C. Cseh, W. Franz, R. Eberhardt "Performance of Stored Geocast", *Proceedings 2003 IEEE 58th Vehicular Technology Conference* Orlando, Florida USA Volume 5 of 5. 6–9 October 2003.

[36] C. Maihofer, W. Franz, R. Eberhardt "Stored Geocast", *Proc. Kommunikation in Verteilten Systemen (KiVS)*, Leipzig, Germany, Feb. 2003, Springer Verlag, pp. 257–68.

[37] F. Kuhn, R. Wattenhofer, Y. Zhang, A. Zollinger "Geometric Ad-Hoc Routing: Of Theory and Practice", *in Proc. 22 nd ACM Int. Symposium on the Principles of Distributed Computing (PODC)*, 2003.

[38] "http://grouper.ieee.org/groups/802/11/".

[39] P. Mohapatra, C. Gi, J. Li "Group Communications in Mobile Ad Hoc Networks", *IEEE Computer, Feb 2004*, pp. 52- 59.

[40] J. Tian, K. Rothermel "Building Large Peer-to-Peer Systems in Highly Mobile Ad Hoc Networks: New Challenges?", *Technical Report 2002/05* , University of Stuttgart, 2002.

[41] "Spatial Aware Geographic Forwarding for Mobile Ad Hoc Networks", *Proc. MobiHoc*, Lausanne, Switzerland, Jun. 2002.

[42] D. Couto and R. Morris. "Location Proxies and Intermediate Node Forwarding for Practical Geographic Forwarding" , *MIT Laboratory for Computer Science technical report MIT-LCS-TR-824*, June 2001.

[43] Sung-Ju Lee, W. Su, M. Gerla "On-Demand Multicast Routing Protocol in Multihop Wireless Mobile Networks", *ACM/Kluwer Mobile Networks and Applications*, 2000.

[44] H. Füßler, M. Mauve, H. Hartenstein, M. Kasemann, D. Vollmer "A Comparison of Routing Strategies for Vehicular Ad-Hoc Networks", *in Proceedings of MOBICOM 2002*, Student Poster, 2002.

[45] M. Kasemann, H. Füßler, H. Hartenstein, M. Mauve "A Reactive Location Service for Mobile Ad Hoc Networks", *TR-14-2002, Department of Computer* Science, University of Mannheim, November 2002.

[46] J. Luo, J.-P. Hubaux "A Survey of Inter-Vehicle Communication", *Technical report IC/2004/24*, School of computer and Communication Sciences, EPEL, 2004.

[47] C. Lochert, H. Hartenstein, J. Tian, H. Füßler, D. Hermann, M. Mauve "A Routing Strategy for Vehicular Ad Hoc Networks in City Environments", *in: Proceedings of IEEE Intelligent Vehicles Symposium (IV2003)*, Columbus, Ohio, June 2003. Hrsg.: IEEE. S. 156-161.

[48] J. Moy, RFC 1584, "Multicast Extensions to OSPF", March 1994.

[49] L. Briesemeister, L. Schafers, G. Hommel "Disseminating Messages among Highly Mobile Hosts based on Inter-Vehicle Communication", *IEEE Intelligent Vehicles Symposium*, pages 522-527, October 2000.

[50] Z. Da Chen, HT Kung, D. Vlah "Ad Hoc Relay Wireless Networks over Moving Vehicles on Highways", *the ACM Symposium on Mobile Ad Hoc Networking & Computing (MobiHoc 2001) Poster Paper*, October 2001.

[51] L. Briesemeister, G. Hommel "Integrating Simple yet Robust Protocol Layers for Wireless Ad Hoc Intervehicle Communications", *Communication Networks and Distributed Systems Modeling and Simulation Conference (CNDS)*, pages 186-192, January 2002.

[52] Mohammad Ilyas, "The Handbook Of Ad Hoc Wireless Networks" CRC Press, 2003.

[53] Xue Yang, Jie Liu, Feng Zhao and Nitin H. Vaidya "A Vehicle-to-Vehicle

Communication Protocol for Cooperative Collision Warning", *in IEEE Aerospace Conference Proceedings*, March. 2002.

[54] "http://www.cartalk2000.net"

[55] R. Prakash, R. Baldoni "Architecture for Group Communication in Mobile Systems", in *Proceedings of the IEEE Symposium on Reliable Distributed Systems (SRDS)*, West Lafayette, Pages 235–242, October 20--23, 1998.

[56] Gruia -Catalin Roman, Qingfeng Huang, Ali Hazemi "Consistent Group Membership in Ad Hoc Networks", *in Proceedings of the 23rd International Conference in Software Engineering (ISCE)*, Toronto, Canada, May 2001.

[57] A. Bruce McDonald, T. Znati "A mobility-based framework for adaptive clustering in wireless Ad-Hoc Wireless Networks", *IEEE Journal On Selected Areas in Communications, Vol.17, No.8*, August 1999.

[58] GARMIN Corporation "GPS Guide For Beginners", 2000.

[59] P. Savvaidis, K. Lakakis, J. Ifadis "Organization of Emergency Response After a Major Disaster Event in an Urban Area with the Help of an Automatic Vehicle Location and Control System", *GPS Solutions, 5(4)*, 70-79.

[60] J.F. Kurose, K.W. Ross "Computer Networking – A top down approach featuring the Internet", Addison-Wesley, 2001.

[61] Sung-Ju Lee, Mario Gerla "Wireless Ad Hoc Multicast routing with Mobility Prediction", *ACM/Kluwer Mobile Networks and Applications, special issue on Design and Deployment of Ad Hoc Networks, vol. 6, no. 4, August 2001*, pp. 351-360.

[62] Matt Bishop "Computer Security Art and Science", Addisson-Wesley, 2004.

[63] Qing Xu, Tony Mak, Raja Sengupta "Vehicle-to-Vehicle Safety Messaging in DSRC", *in Proceedings of the first ACM workshop on. Vehicular ad hoc networks*, pages 19–28, ACM. Press, 2004.

[64] J. Yin, T.r ElBatt, G. Yeung, Bo Ryu, S. Habermas, H. Krishnan, T. Talty "Performance Evaluation of Safety Applications over DSRC Vehicular Ad Hoc Networks", *in Proceedings of the first ACM workshop on Vehicular ad hoc*

networks, Philadelphia, PA, USA, pages: 1 - 9, ACM. Press, 2004.

[65] X Hong, M. Gerla, G. Pei, C.-C. Chiang "A Group Mobility Model for Ad Hoc Wireless Networks", *in Proc. of the 2nd ACM/IEEE Int. Workshop on Modeling and Simulation of Wireless and Mobile Systems (MSWiM'99),* pages 53-60, 1999.

[66] Karen H. Wang, Baochun Li "Group Mobility and Partition Prediction in Wireless Ad-Hoc Networks", *in Proceedings of the IEEE International Conference on Communications (ICC 2002),* New York City, New York, USA, April 2002.

[67] Q. Sun, H. Garcia-Molina "Using Ad-hoc Inter-vehicle Networks For Regional Alerts", *Stanford university, Technical Report Peers,* 2004-51.

[68] K. Tokuda "Application of Wireless Technology to ITS", *Oki technical review, Number 3, Volume 68*, September 2001.

[69] D.-M. Chiu, S. Hurst, M. Kadansky, J. Wesley "TRAM: A Tree-based Reliable Multicast Protocol", *Technical Report TR-98-66, Sun Microsystems*, July 1998.

[70] B. Levine, J. Garcia-Luna-Aceves "A Comparison of Known Classes of Reliable Multicast Protocols", *Proc. International Conference on Network Protocols (ICNP-96),* Columbus, Ohio, Oct 29--Nov 1, 1996.

[71] B. Wang and C.-J. Hou, "Multicast routing and its QoS extension: problems, algorithms, and protocols", *IEEE Network, vol. 14, no. 1, pp. 22 --36*, Jan/Feb 2000.

[72] A. Striegel, G. Manimaran, Iowa State University "A Survey of QoS Multicasting Issues", *in IEEE Communications Magazine, pages 82 --87*, June 2002.

[73] D. R. Cheriton and S. Deering, "Host Groups: A Multicast Extension for Datagram Internetworks", *in Proc. of Data Communication Symposium*, 1985, pp. 172–79.

[74] "http://www.path. berkeley.edu/vanet".

[75] T. Schaffnit "Vehicle Safety Communications in North America", *Annual*

United States Crash Statistics. According to theUnited States National Highway. September 3, 2003.

[76] M. Bechler, W. J. Franz, L. Wolf "Mobile Internet Access in FleetNet", *13. Fachtagung Kommunikation in Verteilten Systemen (KiVS 2003), Leipzig* , February 2003. Ph.D. Thesis, University of Washington, Seattle, WA, 1994.

[77] H. Wu, R. Fujimoto, G. Riley "Analytical Models for Information Propagation in Vehicle-to Vehicle Networks", *in IEEE VTC* 2004-Fall, (2004).

[78] T. Ernst, K. Uehara "Connecting Automobiles to the Internet", *Proceedings 3rd International Workshop on ITS Telecommunications (ITST)*, November 2002.

[79] A. Meissner, L. Wolf, M. Hollick, R. Steinmetz "Security Issues in Group Integrity Management for Multimedia Multicasting", *Proceedings of Third International Conference on Information, Communications & Signal Processing (ICICS 2001)*, Singapore.

[80] A. Mishra, K. M. Nadkarni "Security in Wireless Ad Hoc Networks", *The handbook of ad hoc wireless networks, CRC Press, Inc.*, Boca Raton, FL, 2003.

[81] Q. Huang, C Lu, G.-C. Roman "Reliable Mobicast via Face-Aware Routing", *INFOCOM 2004*, Hong Kong, China, March 2004.

[82] Yi-Bing Lin, I. Chlamtac "Wireless and mobile Network Architectures", *John Wiley and Sons., inc*. 2001.

[83] Vehicle-to-Vehicle Communication - Accident-Free Driving "Calling All Cars", *DaimlerChrysler, HighTech report*, 2001.

[84] B. Maisseu – Renault, ADASE – Advanced Driver Assistance Systems in Europe "IVHW : An Inter-Vehicle Hazard Warning system developed within the DEUFRAKO Programme", *ADASE 2 workshop Paris 2003* , page 2, Paris, 2003.

[85] J. Wu, I. Stojmenovic "Ad Hoc Networks", *Guest editor's introduction, IEEE Press* , 2004.

[86] M. Raya, J.-P. Hubaux "Security Aspects of Inter-Vehicle Communications" *Conference paper STRC - Swiss Transport Research Conference* - 2005.

[87] J.-P. Hubaux, S. Capkun, J. Luo "The Security and Privacy of Smart Vehicles", *Proceedings of the 24th Annual Conference of the IEEE Communications Societies (INFOCOM'05), IEEE Security and Privacy*, May/June 2004.

[88] Filippo Barsotti, Stefano Chessa, Antonio Caruso "The Localized Vehicular Multicast Middleware: a Framework for Ad Hoc Inter-Vehicles Multicast Communications", University of Pisa, submitted to *The Second ACM International Workshop on Vehicular Ad Hoc Networks (VANET 2005).*