A Survey of Dependability Issues in Mobile Wireless Networks

Claudio Basile, Marc-Olivier Killijian, David Powell
cbasile@uiuc.edu, {marco.killijian, david.powell}@laas.fr

February 21, 2003

Technical Report, LAAS CNRS Toulouse, France 2003

Contents

1 Introduction

Mobile wireless networks can be classified in two major categories: *cellular networks* (also known as *infrastructured networks*) and *ad hoc networks*. While cellular networks are characterized by having fixed and wired gateways (*base stations*), which are responsible for routing the messages, ad hoc networks have no fixed infrastructure and all nodes are capable of movement, which determines the network connectivity. Ad hoc nodes can communicate directly only with the nodes that are immediately within their transmission range. To communicate with the other nodes, an intermediate node is used to forward the packet from the source toward the destination. Therefore, in ad hoc networks, nodes need to cooperate in order to maintain connectivity and each node may act as a router. In the sequel, we will focus on ad hoc networks.

1.1 The Characteristics of Mobile Ad Hoc Systems

The main characteristics of ad hoc systems are that they are self-organizing, fully decentralized and highly dynamic. If these characteristics limit the applicability of models and systems built for the wired networks, on the other hand they provide opportunities for a range of new and interesting applications: conferences, meetings, wireless communication between vehicles in road traffic, disaster relief, rescue missions, and battlefield operations. Such scenarios typically lack a central administration or wired infrastructure and, hence, ad hoc systems are particularly appealing for them. In the following paragraphs, we discuss the major challenges in designing systems based on mobile ad hoc systems [1].

Networking. Wireless communication is much more difficult to achieve than wired communication because the surrounding environment interacts with the signal, blocking signal paths and introducing noise and echoes. As a result, wireless connections are of lower quality than wired connections:

1. Lower bandwidths[1];
2. Higher error rates;
3. More frequent spurious disconnections.

These factors can in turn increase communication latency due to retransmission, retransmission timeout delays, error control protocol processing, and short disconnections. Moreover, mobile hosts can move independently from each other,

[1]Cutting-edge products for portable wireless communications achieve from 9.6 Kbps to 4 Mbps (IrDA) for infrared communication, from 1 to 11 Mbps (802.11b) and from 6 to 54 Mbps (802.11a) for radio communication , and 9–14 Kbps for cellular telephony, while Ethernet provides 10 Mbps, Fast Ethernet and FDDI 100 Mbps, ATM 155 Mbps, and Myrinet 2 Gbps. Moreover, for the broadcast nature of wireless communication, the bandwidth availability per user is dependent on the number of users communicating in that area.

which adds unpredictability to the network topological changes. Indeed, connectivity among devices is determined by their relative distance and, so, by their movement.

Most of today's systems have been designed to be operated in a wired environment where network unavailability (or large bandwidth degradation) represent more exceptional situations than a peculiarity of the network itself. As a result, their protocols cannot be used in the mobile environment.

A relevant example is given by TCP which, while being one of the most popular and widely used end-to-end protocols for the Internet, performs poorly in the wireless environment. This is because the assumptions under which TCP was designed do not hold for wireless networks. In particular, TCP considers network congestion to be the primary cause of packet loss and relies on measuring the round-trip time (RTT) and packet loss to conclude if congestion has occurred in the network. In addition, TCP assumes that nodes in the route are static and only performs flow control and congestion avoidance at the source and the destination nodes. However, mobility of nodes in a connection can result in packet loss and long RTT (while the route to the destination is repaired). TCP misinterprets these as due to network congestion and, so, reduces its transmission window size and initiates the slow start phase, where the sending data rate is increased slowly, which significantly reduces unnecessarily communication throughput performance [2].

Several proposals have been made to extend TCP for supporting the ad hoc environment. TCP-F allows the source to be informed of a route disconnection as a result of node mobility. Upon receiving such a notification, it enters a SNOOZE state, in which it suspends data transmission, freezes its timers, congestion window size, and values of other state variables until a route repair message is received. At that time, data transmission is resumed and all timers and state variables are restored as they were before the disconnection [3]. Another approach is given by TCP-BuS, which is described in [4].

The concept of a client initiating service requests to a server for execution and awaiting results to be returned may not be reasonable due to limitations in bandwidth and power. Indeed, networking paradigms need to move toward asynchronous operations, e.g., preferring prefetching and lazy write-back schemes to RPC. In disconnected operation (i.e., a mobile host may intentionally decide to disconnect itself from the network and work only locally), the traditional notion of strong consistency may have to be modified to become less restrictive. Consistency may have different levels and, in particular, there may need to be tolerance of some bounded inconsistency.

Perhaps the concept of remote programming as used in mobile agents is more applicable since it may reduce the interactions exchanged between the client and server over the wireless media. For instance, while in a conventional routing protocol the control information exchanged can be large and result in limited scalability of the algorithm, agent-based routing protocols limit the amount of network messages only to that necessary for agents' migration. The idea is to have agents to perform local computation on some network nodes and then migrate the agents

so as to spread the results of their computations (e.g., a path discovery) over the network. This contrasts with conventional routing protocols where each node performs a local computation on the control information disseminated through the network.

Unlike single wireless link failures, partitioning of ad hoc networks may be due to a large-scale topological change, attributed to the correlated movements of one or more groups of nodes. By capturing the essential characteristics that represent such correlated mobility patterns, one can derive information about the changing network topology and, therefore, be able to predict future network partitioning events for the purpose of building more stable end-to-end connections (see § 2).

Mobile Device Limitations. The implications of portability for mobile devices are small size and weight, and dependence on battery power. Small size and weight means restricted memory size, small storage capacity, and a limited user interface (both data entry and data display). Various techniques can be used to cope with the problems of limited memory such as compressing file systems, compressing virtual memory pages, accessing remote storage over the network, using interpreted script languages instead of compiled object codes, since compiled object codes can occupy more space. General Magic's Telescript and Apple's NewtonScript are examples of such languages.

Batteries are among the largest sources of weight in a mobile node. While reducing battery weight is important, too small a battery can undermine portability, requiring users to have to recharge frequently, carry spare batteries, or use the mobile host less. Power consumption is proportional to CV^2f, where C is the capacitance of the devices and inter-device connections, V is the voltage swing, and f is the clock frequency. Power can be saved by (1) increasing the VLSI integration level so as to reduce C, (2) redesigning chips to operate at lower voltage V, and (3) reducing clock frequency dynamically in order to trade off computational speed for power saving.

Power can be conserved by efficient operation as well. Power management software can power down individual components when these are idle. Applications can conserve power by reducing the computational and communication needs. Wireless transmission, reception, retransmission, and beaconing operations all consume power. Many existing routing protocols use periodic transmission of route update messages to maintain the accuracy of routing tables. In wireless networks, beaconing can also be used to sense the presence of neighboring nodes and then indicate the spatial, temporal, connection, and signal stability of these nodes. Hence, the power consumed as a result of beaconing and its impact on existing applications need to be limited.

System Properties. The principal properties [5] to maintain when designing a robust mobile systems can be summarized as follows:

- *Availability.* Availability represents the survivability of the network services

despite failures even in presence of security attacks (e.g., denial of service attack). The solutions to provide availability in traditional distributed systems have to face, in ad hoc networks, additional complications due to node mobility. On the other hand, denial of service attacks are favoured by the intrinsic broadcast nature of communication in a mobile network.

- *Confidentiality*. Confidentiality measures the absence of unauthorized disclosure of information. One solution can be a public key infrastructure, which can offer integrity and nonrepudiation. In a public key infrastructure, each node has a public-private key pair. Also a trusted third party, i.e., a *Certification Authority*, for key management is needed or the keys have to be delivered in advance. Pre-delivered keys may be preferable because in ad-hoc networks the usage of a single service point is not viable. The service may be replicated, but this is not an easy task in partitionable environment such as ad hoc networks. Moreover, use of asymmetric encryption may be limited by the computational capacity of the mobile host.

- *Integrity*. Integrity is the absence of improper system state aleration such as a corrupted message being transmitted. A message could be corrupted because of benign failures, or because of a malicious attack on the network.

- *Security*. Security is the concurrent existence of availability for authorized users only, confidentiality and integrity with ”improper” meaning ”unauthorized”. There is a number of threats to security in ad hoc systems, as there is a number of safeguards and countermeasures; the reader is referred to [6] for these issues.

Context Awareness. In wired networks, the classical approach to communication is to hide the underlying communication layer, to which most of the error handling would be left (recall that for wired networks, link and host failures are considered to be mostly rare events).

In mobile networks, the role of the network in an application infrastructure is predominant and network events like partitioning and host mobility cannot be handled in a transparent manner (in the operating system or in the middleware, for example) without limiting the scope of the potential applications. Therefore, an opposite tendency has emerged, i.e., to expose the mobile network events to the application, which must be responsible for dealing with them.

Pushing this idea further, the applications become aware of the surrounding environment in which they run and be capable of adapting to it. *Context-aware* computing is a mobile computing paradigm in which applications can discover and take advantage of contextual information such as hardware resources, user location, nearby people and devices, and user activity [7].

1.2 Sensor Networks

Sensor networks constitute a particular kind of ad hoc network. Sensing of environmental data is achieved by the collaborative effort of a large number of sensor nodes, which consist of sensing, data processing, and communicating components. Sensor nodes are typically low-cost, low-power, and small, and can communicate over short distances. They can be densely deployed either inside the phenomenon one aims to sense or very close to it. They can be randomly deployed in inaccessible terrains, e.g., for disaster relief operations, hence, network protocols and algorithms must possess self-organizing capabilities. Another characteristic of such sensor nodes is that they are capable of performing a limited local computation, which can be used for data fusion in order to limit the communication requirements. Sensor networks differ from ordinary ad hoc networks[2] in the following points:

1. The number of sensor nodes in a sensor network can be several orders of magnitude higher.
2. Sensor nodes are densely deployed.
3. Sensor nodes are intended to be very small and, hence, are very limited in power, computation capacities, and memory.
4. Sensor nodes are prone to failures (hardware must be cheap and the sensor life is typically confined to the battery duration).
5. Sensor nodes may not have global identification. As a consequence, location-based routing (a message is routed to a specified geographic area) or, in general, attribute-based routing (a message is routed according to the message contents, as described by attributes included in the message itself) are preferred.
6. Sensor mobility may be limited.
7. Sensor networks are queried from an external user, which may be interested only in some of the data they can provide (e.g., data corresponding to a given geographical location). On the contrary, each node in an ordinary ad hoc network may represent an individual user and, hence, the interaction among nodes tends to be more peer-to-peer.

While traditional networks aim to achieve high quality of service (QoS) provisions, sensor network protocols must focus primarily on power conservation. In fact, sensor power sources are, generally, irreplaceable since sensor nodes, once deployed, are usually inaccessible. Moreover, power is also a scarce resource due to the sensor node's size limitations. As a consequence, sensor network protocols

[2] By ordinary ad hoc network, we mean current ubiquitous computing ad hoc networks.

must have trade-off mechanisms that give the end user the option of prolonging network lifetime at the cost of lower throughput or higher transmission delay. The reader is referred to [8] and [9] for further discussion on sensor networks, and to [10] for biomedical sensor networks.

1.3 Roadmap to this Survey

This survey presents an overview of the work on dependability in the context of mobile ad hoc networks. The attention is on availability and reliability issues. The rest of this article is organized as follows.

Possibly arbitrary node movement makes the network topology very dynamic as well as stochastic; moreover, it can result in frequent network partitioning. This is not the scenario for which most of the network layers for wired networks are build and is also where many of the difficulties for ad hoc networking come. If one could predict the future link availability, then the network layer could exploit such information to take action in advance. For this purpose (and not only for it) several mobility prediction models have been proposed. These models are discussed in § 2.

In recent years, a variety of routing protocols targeting specifically the ad hoc environment have been developed. These protocols cannot be considered dependable as they do not provide consistency guarantees in case of failures or node movement (e.g., atomicity, total order). Yet they constitute the basic primitives on which most of the other higher-level protocols are built and, so, are discussed in § 3.

Mobile computing is not, strictly speaking, a new computing paradigm (the literature on unicast routing in ad hoc networks is fairly voluminous, for instance); nevertheless, dependable mobile computing probably can be considered as such since, since many issues in system modeling, problem definition, and algorithmic solutions are still open. This issues and the proposed solutions are discussed in § 4.

Finally, in § 5 the survey concludes by revisiting the open problems of the field of dependable ad hoc networks and by suggesting new directions.

2 Mobility Models

Researchers have proposed many mobility prediction schemes to predict the future availability of wireless links [11], [12], [13]. These models have been used (1) to describe how mobile hosts move so as to evaluate the performance of the proposed routing protocols in a realistic representation of the scenario in which the protocol would be actually used, and (2) to predict the network connectivity during a routing protocol execution for the purpose of building more stable end-to-end connections.

Historically, the first mobility models used for ad hoc networks were variations of the *random walk* model, which defines individual node movements and is based on random directions and speeds. These models have had a limited success in describing realistic situations. Indeed, in reality, mobile users often exhibit cor-

related mobility patterns in their movements. Such correlated mobility patterns are also referred to as *group mobility*. In a museum, visitors move at different paces and along different routes depending on their varied interests, but their mobility patterns tends to be focussed on common points of interests, such as a painting. In *collaborative* computing environments , the mobile users do not behave randomly, but are involved in team activities in which they perform common tasks (a group of firefighters on a disaster scene) or have similar destinations (visitors heading for similar objects of interest).

The grouping behavior of the mobile users has been observed in actual field trials of local area wireless networks [14]. In [15], several representative group-based user mobility patterns existing in different ad hoc network scenarios are identified.

One can further observe that the group-based node movements cause the network to partition. Consider an ad-hoc network consisting of many movement groups whose nodes are initially dispersed and intermixed, the distinct mobility patterns of each group cause the groups to split, and the network eventually partitions. For a fully-connected network to partition into completely disconnected components, such large-scale and structured topology changes can only be caused by correlated movements of a group of nodes, whereas independent movement of individual nodes can only cause random and sporadic link breakage. This insight agrees with the simulation results from [15] and [16], which have shown that the group mobility behavior of mobile users causes frequent network partitioning, and the resulting partitions are the separate mobility groups.

One of these group mobility models is called *Reference Point Group Mobility (RPGM)* model [15]. In this model, the nodes in the network are organized into mobility groups. Each mobility group has a logical group center, the *reference point*, which defines the movement of the entire group. The RPGM model describes the group membership of a mobile node by its physical displacement from the group's *reference point*. For example, at time t, the location of the node i in the group j is given by the following location vectors:

1. Reference location $\mathbf{Y}_j(t)$;
2. Local displacement $\mathbf{Z}_{ji}(t)$;
3. Node location $\mathbf{X}_{ji}(t) = \mathbf{Y}_j(t) + \mathbf{Z}_{ji}(t)$.

The RPGM model generates the physical locations of the mobility nodes, but it may not be used to accurately identify mobility groups. For example, consider a network topology generated by the RPGM model where there are several mobility groups with common reference points and with overlapping coverage areas. Since in this scenario the member nodes are all intermixed, it is impossible to recognize the mobility groups based only on the node physical location. Since the nodes exhibit grouping behavior in their movements, a more distinguishing characteristic of nodes within the same mobility group is the *node velocity*. In other words, the mobility patterns are correlated based on the velocity of nodes.

Wang and Li [17] extend the RPGM model and propose a *Reference Velocity Group Mobility (RVGM)* model. In this model, each mobile node is represented by its velocity vector; each mobility group has a characteristic *mean group velocity*, which each member node's velocity slightly deviates from. The membership of node i in the group j is described by the addition of two velocity vectors:

1. Mean group velocity $\mathbf{W}_j(t)$;
2. Local velocity deviation $\mathbf{U}_{ji}(t)$;
3. Node velocity $\mathbf{V}_{ji}(t) = \mathbf{W}_j(t) + \mathbf{U}_{ji}(t)$.

The node velocity in each mobility group is modeled in [17] by a Gaussian distribution parametrized by the mean group velocity and a variance representing the amount of variation in the member node velocities.

Camp et al. [18] provide a survey on mobility models for ad hoc networks. They simulate the Dynamic Source Routing (DSR) protocol [19] using different models. In particular, their results show that the performance of an ad hoc network protocol can vary significantly (1) with different mobility models and (2) with the same mobility model used with different parameters.

3 Ad Hoc Routing Protocols

Due to the limited transmission range of wireless network interfaces, multiple network hops may be needed for one node to exchange data with another node across the network. In recent years, a variety of new routing protocols targeting specifically the ad hoc environment have been developed. As these protocols constitute the basic primitives on which most of the other higher-level protocols are built, in the next sections we discuss them in some detail.

3.1 Unicasting

Unicasting protocols for ad-hoc routing may generally be categorized as:

1. *Topology-based* routing protocols. These protocols use the information about links in the networks to perform packet forwarding and can be further divided into:
   (a) *Proactive* protocols (e.g., DSDV, CGSR presented in § 3.1.1), in which nodes periodically refresh the routing information so that every node always has consistent, up-to-date routing information from each node to every other node in the network.
   (b) *Reactive* protocols (e.g., DSR, AODV, TORA, ABR, SSR presented in § 3.1.2), where the routing information is propagated to a node only when it is necessary, i.e., when the node requests it.

(c) *Hybrid* protocols (e.g., ZRP presented in § 3.1.3), which make use of both reactive and proactive approaches so as to incorporate the merits of both of them.

The reader is referred to [20], [21] and [22] for a survey and a comparison of the topology-based approaches.

2. *Position-based* routing protocols (e.g., LAR, Terminodes presented in § 3.1.4). These protocols aim to surpass some of the limitations of topology-based protocols by using additional information, i.e., the physical location of nodes.

Current ad hoc routing approaches have also introduced new paradigms, such as power awareness, routing in disconnected ad hoc networks and agent-based routing. These will discussed in the following sections as well.

3.1.1 Proactive Protocols

In proactive protocols, each node maintains one or more tables to store routing information. This information is kept up-to-date by means of periodic message exchanges. The areas in which these protocols differ are the number of necessary routing-related tables and the methods by which changes in network structure are broadcast. The main drawback of these protocols is that the maintenance of unused paths is an unnecessary waste of resources. If the network topology changes frequently, then a significant part of the bandwidth may be occupied.

The *Destination-Sequenced Distance-Vector (DSDV)* routing protocol [23] is based on the Bellman-Ford routing mechanism [24], which has been improved to avoid loops in the routing tables. Every node in the mobile network maintains a routing table for all the possible destinations within the network. Routing table updates are periodically transmitted throughout the network in order to maintain table consistency. Two possible kinds of packets are used: *full dump* packets, which carry all available routing information and, so, can require multiple network protocol data units (NPDUs), and *incremental* packets, which relay only the information that has changed since last full dump.

The *Clusterhead Gateway Switch Routing (CGSR)* protocol [25] is based on DSDV but imposes a clustered structure to the network and uses several heuristic routing schemes. In each cluster, a node is elected as cluster head by running a cluster head selection algorithm. A packet sent by a node is first routed to its cluster head, then the packet is routed from the cluster head to the gateway node of an adjacent cluster (a node at the border of the two clusters). The gateway node then routes the packet to the cluster head of the adjacent cluster, and so on until the cluster head of the destination cluster and, thence, the actual destination node. CGSR uses DSDV as its underlying routing protocol and, hence, has a similar overhead.

3.1.2 Reactive Protocols

Reactive protocols take a different approach as they create routes only when desired by the source node. When a node requires a route to a destination, it initiates a *route discovery* process, which is completed once a route is found or all possible route permutations have been examined. Since routes are discovered only on demand, the first packet to be transmitted will likely suffer from a large delay. Once a route has been established, it is maintained by a *route maintenance* procedure until either the destination becomes inaccessible along every path from the source or until the route is no longer desired.

The *Dynamic Source Routing (DSR)* protocol [19] is based on the concept of source routing. When a node receives a route request packet (containing both the source and the destination node addresses) it checks whether it already knows a path to the destination and, if not, adds its own address to the *route record* contained in the packet and forwards the packet along its outgoing links. A *route reply* is generated when the route reaches either the destination or an intermediate node that has a route to the destination. This packet contains the whole route to the destination and is sent back to the initiator, passing through all the nodes indicated by the route record previously formed. A disadvantage of the protocol is that it suffers from a scalability problem due to the nature of source routing. As the network becomes larger, control packets (which collect node addresses for each node visited) and message packets (which contain full source routing information) also become larger. Clearly, this has a negative impact due to the limited available bandwidth.

The *Ad Hoc On-Demand Distance Vector (AODV)* routing protocol [26] builds on the DSDV algorithm and minimizes the overhead of the latter by creating routes only on demand. Instead of source routing, AODV relies on dynamically establishing route table entries at intermediate nodes. The path discovery process is initiated by a node by broadcasting a route request packet (RREQ) to its neighbors, which then forward the request to their neighbors, and so on, until either the destination or an intermediate node with a route to the destination is located. By the time a RREQ packet reaches the destination or a node that can supply a route to the destination, a reverse path has been established to the source of the RREQ. A unicasts route reply packet (RREP) travels back to the source permitting each node along the path to set up a forward pointer to the node from which the RREP comes. Routes are maintained as follows. If a source node moves, it can reinitiate the route discovery protocol; if a node along the route moves, its upstream neighbors notice the move and propagate a link failure notification message to their upstream neighbors and so on up to the source, which may decide to reinitiate the route discovery protocol.

The *Temporally Ordered Routing Algorithm (TORA)* [27] is based on the concept of link reversal and aims to operate in a highly dynamic mobile networking environment. The key design concept of TORA is the localization of control messages to a very small set of nodes near the occurrence of a topological change. Each node has associated a “height” metric, which is used to establish a directed acyclic

graph (DAG) rooted at the destination. Links are assigned a direction based on the relative height metric of neighboring nodes. Timing is an important factor for TORA as the height metric depends on the time of a link failure. Indeed, TORA assumes that all nodes have synchronized clocks (accomplished via an external time source such as the Global Positioning System).

Conceptually, the quintuple $(\tau_i, oid_i, r_i, \delta_i, i)$ representing the height of a node i is defined by two parameters: a reference level (the first three values) and a delta with respect to the reference level (the last two values). When a node i loses its last downstream link (due, for example, to a link failure), it generates a new reference level by using the reference levels propagated by its neighbors. The first value representing the reference level, τ_i, is the time of the link failure. The second value, oid_i is the unique ID of the node that defined the new reference level and is used to ensure that reference levels can be totally ordered lexicographically, even of multiple nodes define reference levels due to failures occurring simultaneously. The third value, r_i, is a single bit used to divide each of the unique reference levels into two unique sublevels. This bit is used to distinguish between the original reference level and its corresponding, higher reflected reference level. The first value representing the delta, δ_i, is an integer used to order nodes with respect to a common reference level. This value is used in the propagation of the reference levels. Finally, the second value representing the delta, i, is the unique ID of the node itself. When a new reference level is generated, links may be reversed to adapt to the new reference level.

The *Associativity-Based Routing (ABR)* protocol [28] uses the *degree of association stability* as a metric. Each node periodically generates a beacon to indicate its presence to neighbor nodes. For each beacon received, the associativity counter (called "associativity tick" in [28]) of the current node with respect to the beaconing node is incremented. Associativity counters are reset when the neighbors of a node or the node itself move out of proximity. Longer-lived routes are preferred since they indicate less node mobility and, so, more stability. Route discovery is accomplished by a broadcast query and wait-reply (BQ-REPLY) cycle. All nodes receiving the query message append their addresses and their associativity counters with their associated neighbors to the query packet. A successor node erases its upstream node neighbor's associativity counter entries except the one concerning itself. As a result, each packet arriving at the destination will contain the associativity counters of the nodes along the route from the source to the destination. The destination selects the "best" route by examining this information and sends a REPLY packet back to the source along the chosen path.

The *Signal Stability-Based Adaptive Routing (SSR)* protocol [29] selects routes based on the signal strength between nodes and a node's location stability in order to choose routes that have stronger connectivity. The signal strength is obtained by periodic beacons from the link layer of the neighbor nodes. During the route discovery process, route requests are forwarded to the next hop only if they are received over strong channels and have not been previously processed. The destination chooses the first arriving route-discovery packet because it is most probably

the packet arrived from the shortest and/or least congestioned path; moreover, it must be a path of strong signal stability, as the packets are dropped at a node if they arrive from a weak channel. The route is then reversed and a route-reply message is sent back from the destination to the initiator along the chosen route.

3.1.3 Hybrid Protocols

Hybrid ad-hoc routing protocols combine local proactive routing and global reactive routing in order to achieve higher efficiency and scalability.

In the *Zone Routing Protocol (ZRP)* [30] a route discovery is initiated on demand. A *routing zone* is defined for each node and includes the nodes whose distance is less than a predetermined maximum distance (each node specifies a zone radius in terms of radio hops). A routing zone is similar to a cluster with the exception that zones can overlap and, hence, every node acts both as a cluster head and as a member of other clusters. Each node is required to know the topology of the network within its zone only. Updates about changes in topology within the zone are propagated by using a proactive routing protocol. Each node, therefore, has a route to all other nodes in the same zone. If the destination node resides outside the source zone, a reactive search-query routing method is used.

3.1.4 Position-based Protocols

Position-based protocols require that information about the physical position of the ad hoc nodes is available. Each node may determine its own position through the use of a Global Position System (GPS) or some other type of positioning service (a survey of these methods can be found in [31]). A *location service* may be used by the sender of a packet to determine the position of the destination so to include it in the packet. The routing decision at each forwarding node is then based on the destination's position contained in the packet and the position of the node's neighbors. Position-based routing does not necessarily require the establishment or maintenance of routes. As a further advantage, position-based routing supports the delivery of packets to all nodes in a given geographical region. This type of service is called *geocasting* and is discussed in § 3.3. A survey on position-based routing protocols can be found in [32].

The *Location-Aided Routing (LAR)* protocol [33] utilizes location information to improve performance. The search for a new route is limited to a small request zone, thus reducing the signaling traffic. LAR assumes that the sender has knowledge of the destination location and velocity. Based on this information, the destination *expected zone* can be defined. The *request zone* is the smallest rectangle including both the location of the sender and the destination expected zone. The sender explicitly specifies the request zone in its route request message. Nodes that receive the request message but are outside the request zone discard the packet.

The *Terminodes* project [34] combines hierarchical and position-based routing in a two-level hierarchy. Packets are routed according a proactive distance vector

scheme if the destination is close to the sending node. For long-distance routing, a greedy[3] position-based approach (called *Anchored Geodesic Packet Forwarding*) is used. Once a long-distance packet reaches the area close to the recipient, it continues to be forwarded by means of the local routing protocol. In order to prevent the greedy forwarding used for long distance from getting trapped into a local minimum, the sender includes a list of positions (*anchors*) in the packet header. The packet must then traverse the areas at these positions on its way to the sender. The packet forwarding between these areas is done on a purely greedy basis. Therefore, this approach is a form of position-based source routing, as the sender needs to know about the appropriate positions leading to the destination. The sender requests this information from nodes that it is already in contact with. Once the sender has the information, it needs to check at regular intervals whether the path of positions is still valid or can be improved.

Each node is required to know its own position. For the case where a GPS is not available (e.g., the GPS signal may be too weak or jammed, or a GPS solution cannot be afforded for cost or integration reasons), a *Self Positioning Algorithm (SPA)* is proposed in [35]. SPA uses range measurements between the mobile nodes to build a network coordinate system. The *Time of Arrival (TOA)* method [36] is used to obtain the distance between two mobile nodes.

In mobile ad hoc networks, a node may be required to forward packets on behalf of another node. This consumes energy (reduces battery life) without direct advantages. Therefore, if not all mobile nodes belong to the same administration authority (as in military operations, disaster relief, rescue missions), their users may tend to be selfish: they use services provided by others but do not want to provide services to the community. Clearly, selfish nodes may break the functioning of the network completely. Therefore, a stimulation mechanism is necessary to encourage users to provide services to each other. The Terminodes project introduces a virtual currency, called *nuglets*, and a mechanism for charging/rewarding service usage provision. Terminode hardware comes with an initial stock of nuglets, which have no monetary value and can only be used within terminode networks. Terminodes must pay to those terminodes that provide the packet forwarding service. In particular, packet forwarding can be paid by either the originator (*Packet Purse Model*) or the destination of the packet (*Packet Trade Model*).

3.1.5 Power-aware Protocols

Power is a precious and limited resource for wireless ad hoc networks. The focus on battery technology research has been to increase battery power capacity while restricting the weight of the battery. However, unlike other areas of computer technology such as microchip design, battery technology has not experienced significant advancement in the past 30 years [37]. Although hardware-based techniques

[3] Among the algorithms for optimization problems, *greedy algorithms* always make the choice that looks best at the moment, i.e., they make a locally optimal choice in the hope that this choice will lead to a globally optimal solution.

(e.g., low-power design, variable clock speed CPUs, flash memory, disk spindown) have resulted in considerable energy saving, other ways should be explored as well to improve energy efficiency. One possibility is to design the higher layers of the protocol stack of mobile nodes with energy efficiency as an important goal. The general guidelines that may be adopted for an energy-efficient protocol design are the following:

1. Collisions should be eliminated as much as possible within the MAC layer since they result in retransmissions, which in turn lead to additional power consumption and to possibly unbounded delays. The EC-MAC protocol [38] is one example that avoids collisions during reservation and data packet transmission.

2. At the link layer, transmissions may be avoided when channel conditions are poor. Also, error control schemes that combine *Automatic Repeat Request (ARQ)* and *Forward Error Correction (FEC)* mechanisms may be used to trade off retransmissions with ARQ versus longer packets with FEC.

3. Routes should be established so that all nodes equally deplete their battery power, as studied in [39]. Routing through nodes with lower battery power should also be avoided. In [40] the topology of the network is controlled and modified by varying the transmit power of the nodes. The authors formulate a constrained optimization problem with two constraints: connectivity and biconnectivity; and one optimization objective: maximum power used.

4. The operating system should suspend specific subunits (e.g., network, disk, memory, display, etc.) based upon prolonged inactivity. Within the application layer, the power conserving mechanisms tend to be application specific [41].

The reader is referred to [42] for a survey on energy- efficient network protocols for wireless networks.

3.1.6 Disconnected Ad Hoc Routing

An approach to deal with disconnected ad hoc networks is to let the mobile host wait passively for the network to reconnect. This may lead to unacceptable transmission delays for the application. Some works have, therefore, proposed approaches that try to limit these delays by exploiting and controlling node mobility.

Li and Rus [43] address the problem of mobile users that are disconnected in ad hoc networks. In contrast to letting the mobile host wait passively for reconnection, the mobile hosts actively modify their trajectories to minimize transmission delay of messages. Two flavors of their approach distinguish whether the movement of all hosts in the system are known or not known. The system is intended for applications like field operations or disaster relief that require urgent message delivery and involve cars or robots.

Vahdat and Becjer [44] propose an *epidemic* routing protocol for disconnected networks. The routing mechanism is derived from epidemic algorithms that provide eventual consistency in replicated databases without requiring any particular replica to be available at any time. Epidemic routing relies upon carriers of messages coming into contact with another component of the network though node mobility. At this point, nodes exchange pair-wise messages that the other node has not seen yet. Even if there never exists a path in the momentary snapshot of the network, the transitive transmission of data eventually causes a message to reach its destination. Their simulation results show that, in the scenarios considered, epidemic routing is able to deliver nearly all transmitted messages while existing ad hoc routing protocols fail to deliver any messages because of the limited node connectivity. The required buffering, of course, causes increased resource consumption.

Chatzigiannakis et al. [45] present a *snake* protocol, where a snake-like sequence of carriers (called *support stations* in the paper) always remain pairwise adjacent and move in a way determined by the snake's head. The head moves by executing a random walk over the area covered by the network. The protocol is theoretically analyzed. Results derived from an implementation show that only a small number of carriers is required for efficient communication.

Chatzigiannakis et al. [46] extend the work in [45] by presenting a new protocol, called the *runners*, where each carrier performs a random walk sweeping the whole area covered by the network. The authors perform an experimental evaluation and comparison between the *snake* protocol and the *runners* protocol. It turns out that the *runners* protocol is more efficient (smaller message delays and memory requirements) and robust than the *snake* protocol. The authors also note that while the snake protocol is resilient only to one carrier failure, the runnur protocol is resilient to up to $N - 1$ failures, where N is the number of carriers.

3.1.7 Agent-Based Ad Hoc Routing

Amin and Mikler [47] propose an *Agent-based Distance Vector Routing (ADVR)* algorithm. For each round, the number of messages exchanged in the network is bounded by the number of the agents present in the network. A route discovery manifests in the movement of agents carrying routing information from one node to another node. To reduce the amount of information propagated, agents refrain from transferring complete routing tables whenever possible. Instead, agents identify routing table entries that have been modified but yet to be transferred to a particular neighbor. The migration strategy employed uses a combination of *Stigmergy*, as a form of indirect communication, and a depth-first-search. Stigmergy is a mechanism that insects use to communicate with each other by changes in the environments. The agents indicate their presence using pheromone trails (a pheromone is a volatile chemical substance released by an animal in the environment and serves as a stimulus to other individuals of the same species for one or more behavioral responses). While ants use pheromone trails to follow the path of

the successor ant, in ADVR pheromone tracks of one agent repel other agents. An agent traversing a link from node x to node y deposits a pheromone on the link. Another agent migrating from x will choose a link with the weakest pheromone value thereby migrating to a least recently visited region of the network.

3.2 Unreliable Broadcasting and Multicasting

In addition to the above work on unicast routing in ad hoc networks, there has been significant work on unreliable broadcasting and multicasting as well, and several protocols have been proposed. These protocols are unreliable in the sense that because a message may be lost if the network topology changes during the multicasting of the message, i.e., no guarantees on message delivery is provided for partitionable networks.

Williams and Camp [48] provide a classification and comparison of these approaches. Four principal families are so distinguished: (1) *Simple Flooding*, where a source node broadcasts a packet to all neighbors, each of which broadcasts in turn the packet to its neighbors—this is done only if the packet was not already forwarded; (2) *Probability Based Methods*, which are similar to flooding except that nodes only forward with a probability determined by their perception of the network topology; (3) *Area Based Methods*, where a node refrains from forwarding a packet received from another node if the additional area that would be so covered is too low; and (4) *Neighbor Knowledge Methods*, where each node maintains state on its neighbors so to avoid unnecessary forwarding.

Zhou and Singh [49] propose a *Content Based Multicast (CBM)* scheme for ad hoc networks. In CBM, the content of the data being multicast together with the mobility of the receivers determine the multicast set. The authors focus on battlefield applications but mention also the possibility of use in disaster relief. The CBM protocol is based on the idea of "sensor-push" and "receiver-pull". Sensors detecting threats push the information out into the network to some distance and direction. Individual receivers then pull threat warnings from nodes that lie in the direction of their travel. The protocol assumes the area of operation to be mapped and divided into regions. Every node has location capabilities by employing GPS. A leader per region maintains a list of all threat warnings received via push packets. Nodes pulling these threat warnings send a query to the leader of the region that they are traveling to. When the leader leaves its block, the responsibility for maintaining threat warnings passes on to a new leader.

3.3 Geocasting

In geocasting, a variant of the conventional multicasting problem, messages are delivered to all hosts within a given geographical region. In traditional multicasting, a host becomes a member of the multicast group by explicitly joining the multicast group (usually a named entity). On the other hand, a host automatically is a member of a geocast group if its location belongs to the region specified for

the geocast—this region is referred to as *geocast region*. The set of the nodes in the geocast region is said to form a *geocast group*. For a node to be able to determine whether it belongs to a geocast group, the node must be able to derive its own physical location (e.g., by means of GPS). See [50] for a review of geocasting protocols.

4 Fault-Tolerant Algorithms in Ad Hoc Networks

In distributed computing, several recurrent problems have been isolated, such as distributed mutual exclusion, consensus, leader election, distributed commit, group communication. These problems have been identified as central problems to solve as they form the building blocks for solving application-specific problems.

The study of recurrent problems in the context of mobile computing involves two complementary paths. First, problems that are already defined in the context of distributed computing should be adapted to the new domain (i.e., mobile computing), whenever applicable. Second, problems that are specific to the characteristics of the new domain should be identified (e.g., location-dependent problems such as geocasting and location-based group membership service). The identification of the application domains for mobile computing plays also a fundamental role since it enables the identification of generic objectives that must be satisfied. However, it does seem inevitable that protocols for achieving desired system properties despite faults and mobility will often need to be application-specific.

In mobile computing, substantial real applications are still scarce, the formal study of generic problems is quite recent, and the process of formalizing these problems, verifying and comparing the proposed solutions is often, in our opinion, just at an early stage. In the rest of this section, we present some of the most relevant work that has been done in this direction and, in particular, we focus on transactional applications, group communication, leader election, and distributed mutual exclusion problems.

4.1 Transactional Applications on Ad Hoc Networks

Because of mobility, transactional applications in the ad hoc context must cope with the possibility that even normal system operation may lead to violations of the database correctness. As a result, research has focused on redefining the notion of correctness so as to adapt to the new constrains of ad hoc networks. A number of alternative definitions of ACID properties have been identified that weaken one or more of the properties. The general trend is to allow a certain degree of autonomy in transaction processing during disconnections. This is done by permitting bounded inconsistency among the data copies.

For example, in disconnected operation, a database client maintaining a local copy of the most recently used data could continue executing even while being disconnected from the server. User transactions can be decomposed into a number

of *weak* and *strict* sub-transactions according to the degree of consistency required by the application. Strict transactions maintain the traditional notion of transaction and, if committed, are always committed globally. As a result, strict transactions can be committed only while being connected with the server. On the other hand, weak transactions are first committed locally—when the user issues the transaction commit operation—and are used to guarantee a consistent local view of the data.

When connectivity with the served is reestablished, an implicit global commit is performed for committed weak transactions in order to guarantee their durability; however, the application needs to handle the possibility of having transactions that, notwithstanding having been committed locally, may be aborted on performing the global commit [51], [52], [53].

The reader is referred to [51], [52] and [53] for discussion on database consistency, and to [54] for a unilateral commit protocol, in the context of partitionable mobile networks.

4.2 Group Communication on Ad Hoc Networks

Before introducing the work on the group communication problems (reliable and atomic broadcast, group membership, consensus, etc.) over ad hoc networks (see § 4.2.3), we first provide introductory material on the specification of a group membership service for distributed computing (presented in § 4.2.1) and, consequently, on partition-aware applications (presented in § 4.2.2). Our intent is to recall the theoretical impossibilities due to asynchronous communication. The situation becomes even more complicated in mobile systems due to node mobility, which can cause partitioning.

Partitioning is both an intrinsic characteristic of mobile computing and a major obstacle in defining generic solutions for mobile computing (and distributed computing as well). This typically reflects in a conditional definition of the liveness property for a given specification (e.g., the system is required to deliver the service given that network topology eventually "stabilizes" or given that the network topology "stabilizes" infinitely often and for a sufficient amount of time so that the algorithm can make progress). An approach equivalent to expressing a conditional liveness property, is the use of unreliable failure detectors. Although unreliable, these failure detectors must exhibit specific properties that, being not theoretically implementable in partitionable asynchronous systems, implicitly express the necessary additional stability condition.

4.2.1 Group Membership Service Specification

A *group membership* protocol manages the formation and maintenance of a set of processes called a *group*. For example, a group may be a set of processes that are cooperating toward a common task (e.g., the primary and backup servers of a database), a set of processes that share a common interest (e.g., clients that subscribe to a particular newsgroup), or the set of all processes in the system that are

currently deemed to be operational. In general, a process may *leave* a group because it failed, it voluntarily requested to leave, or it is forcibly expelled by other members of the group. Similarly, a process may *join* a group (e.g., it may have been selected to act as a replicate for the other processes in the group). A group membership protocol must manage such dynamic changes in a coherent way: each process has a *local view* of the current membership of the group, and processes in the group need to agree on these local views despite failures [55].

Another well-known problem requiring agreement in spite of failures is *consensus*. This problem cannot be solved deterministically in asynchronous systems even if communication is reliable, only one process may fail, and it can do so only by crashing [56]. Since the purpose of group membership is to ensure some kind of agreement among processes, the potential for running into a similar impossibility result is obvious. On the other hand, depending on how it is specified, group membership is different from consensus in at least two ways: (1) in group membership, a process that is suspected to have crashed can be removed from the group, even if this suspicion is actually incorrect; (2) consensus requires progress in all runs, while group membership allows runs that "do nothing". These differences appear to make group membership weaker than consensus, and in fact (1) has been widely cited in the past as a reason why group membership is solvable in asynchronous systems while consensus is not. However, fot the case of group membership services that aim to maintain a *single* agreed view of the current membership of a group, it has been shown that group membership is not solvable deterministically in asynchronous systems, where communication is reliable and where at most one process may crash [57]. These are called *primary-component* group membership services and are intended for systems with no network partitions, or for systems with strong consistency requirements, which allow the group membership to change in at most one network partition, the "primary component".

To escape this impossibility result, so-called *partitionable* group membership services have also been proposed. These allow *multiple* views of the group to coexist, i.e., different views of the membership of the group may evolve concurrently and independently from each other. In particular, there may be several disjoint subsets of processes such that processes in each subset agree that they are the current members of the group. Such group membership services allow *group splitting* (e.g., when the network partitions) and *group merging* (e.g., when communication between partitions is restored).

However, these partitionable group membership services run into another problem: their specification must be strong enough to rule out useless group membership protocols (in particular, protocols that can capriciously split groups into several concurrent views of the same group or capriciously install new views excluding correct and non-suspected processors) and yet it should be weak enough to remain solvable [58]. These problems have been identified in two papers widely referenced to give rigorous definitions of group membership for asynchronous systems: [59] for the primary-component type, and [60] for the partitionable one. Since the work of [58], several other group membership specifications have appeared. De-

spite this intense activity, the distributed system community has yet to agree on a formal definition of the group membership problem, especially for partitionable systems.

Friedman and Renesse [61] give a specification for the Horus group communication system. Congress and Moshe [62] are two membership protocols that have been designed by the Transis group. Congress provides a simple group membership protocol, while Moshe extends Congress to provide a full group communication service.

Cristian [63] proposes three group membership specification for the *timed asynchronous* model [64] with three different consistency guarantees: *group agreement*, *majority agreement*, *strict agreement*. The group agreement represents a partitionable group membership specification and is described in the following.

It is assumed that a unique service S is implemented by servers replicated on a fixed team of processes P. Processes exchange messages via a datagram communication service. Messages can get lost and communication delays are unbounded, however most messages arrive at their destination within a known one-way *timeout* delay constant d. Thus, the datagram service has *omission/performance* failure semantics.

Processes have access to *stable storage* and *hardware clocks*. Clocks are not required to be synchronized; however, the drift rate of a correct hardware clock is bounded by a priori known constant r. *Crash* failure semantics is assumed for hardware clocks, moreover a non-crashed process has a correct hardware clock. Servers are scheduled to run on processors and the scheduling delays are unbounded; however, most actual scheduling delays are shorter that a known constant s, meaning that a process is likely to react to any trigger event (i.e., a timer event) within s time units. When scheduling delays exceed s, servers suffer performance failures. Thus, servers have *crash/performance* failure semantics. The model assumes that processes do not perform any incorrect state transitions: a process crashes by stopping to execute its program. Any crashed server eventually *restarts*. The higher level worst-case server to server timeout delay is given by $\delta = s + d + s$.

Two processes p and q are *connected* in a time interval $[t, t']$ if they are correct (i.e., non-crashed and timely) and any message sent between them in $[t, t' - \delta]$ is delivered within δ time units. Two processes p and q are *disconnected* in $[t, t']$ if no message sent between them in $[t, t' - \delta]$ is delivered within δ time units, or p or q is crashed in $[t, t']$. Processes p and q are *partially connected* in $[t, t']$ if they are neither connected nor disconnected in $[t, t']$. Note that any pair of processes can only be in one of the above modes. A set of processes that are pairwise connected form a *physical partition*. A timed asynchronous system is *stable* in $[t, t']$ if during this time interval (1) no process fails or restarts, (2) all pairs of processes are either connected or disconnected, and (3) the "connected" relation between processes is transitive. Note that a stable system consists of one or more disjoint physical partitions. It is assumed that the system alternates between long stability periods and comparatively short instability intervals, i.e., synchronous communication can be achieved most of the time.

The membership service groups team members that can communicate among themselves in a timely manner into groups. In the absence of failures, groups include all team members that are not crashed. Transient instability periods, or more permanent disconnection periods between different parts of a network, can result in the creation of several parallel groups. When communication among parallel groups is reestablished and a sufficiently long stability period follows, they can be re-merged into a maximal group.

A group G is a said to be a *sucessor* of group G' if there exists some process p that joined G after joining G'. Two groups G and G' are said to be *parallel* if neither is a successor of the other. Processes p and q are (logically) partitioned at time t if they are joined to different parallel groups at t. A group G is a *majority group* if the set $mem(G)$ of its members contains a numeric majority of the team members P, that is, $|mem(G)| > \frac{|P|}{2}$. The group agreement protocol makes visible to the processes the existence of both majority and minority groups. Roughly speaking, when a new group G is installed at a process p, p is informed of the predecessor group $pred(q, G)$ of each process q that is member of G—$pred(q, G)$ is the previous group to which q was joined before joining G. Initially all processes are joined by definition to a predecessor group $(0, 0)$ with initial state s_o and membership P. The above requirement on the installation of a new group G permits any two members p and q of group G to detect if they were (logically) partitioned before joining G. In such a case, they could have applied conflicting updates to their local states and, so, may have diverged. If state divergence is detected, the initial state of the new group G must reconcile the conflicting updates. This task is application specific. It is assumed the exsistence of a *state merge function* f that reconciles any conflicting updates applied to states s' and s'' to produce a reconciled state s. This functions is used when a new group is formed so as to ensure consistent state among the new group members.

Fekete et al. [65] present a formal specification for a partitionable group communication service. In the same work, the service is used to construct an ordered broadcast application and, in a subsequent work, to construct replicated data services [66]. The specification separates safety requirements from performance and fault-tolerance requirements, which are shown to hold in executions that stabilize to a situation where the failure status stops changing and corresponds to a consistently partitioned system.

Babaoglu et al. [67] give a formal specification and an implementation for a partitionable group communication service in asynchronous distributed systems. The specification and implementation presented form the basis of Jgroup [68], a group-enhanced extension to the Java RMI distributed object model. The asynchronous system model consists of a finite set of processes communicating by exchanging messages. Processes may crash and communication links can transiently fail. The system behaves benignly because it ensures eventual symmetry of reachable (unreachable) processes and fair channels. The proposed *Partitionable Group Membership Service (PGMS)* comprises the properties of View Accuracy,

View Completeness, View Coherency, View Order, and View Integrity. The authors show that for every implementation of PGMS, a run exists that violates a property. Thus, it is impossible to solve PGMS in asynchronous systems. In the next step, failure detectors are employed to detect crashed processes. The failure detector exhibits the two properties of Strong Completeness[4] and Eventual Strong Accuracy[5]. Theoretically, the deterministic implementation of the required failure detector is impossible, too, in asynchronous systems. However, the properties of Strong Completeness and Eventual Strong Accuracy reflect the stability condition of the distributed system if they are satisfied. With such a failure detector, the group membership problem becomes solvable and an implementation is presented. Finally, a reliable multicast service complementing PGMS is specified.

None of the work presented above provides a self-stabilizing solution[6] for the group membership problem. Dolev and Schiller [70] propose a randomized algorithm for implementing self-stabilizing group membership service in asynchronous systems. A randomized self-stabilizing data-link algorithm is used to ensure that a message sent over a link arrives at its destination before the next message is sent. A process' algorithm consists of an infinite loop that includes a communication step with every neighbor process. A process can send a message to all of its neighbors in one single communication operation. Processes may crash and recover during the execution. Although the system is asynchronous, it is assumed that each process eventually knows the set of its non-crashed neighbors. A transient fault detector is used to trigger the update algorithm, which is such that when it stabilizes, processes have a consistent view of their connected component.

4.2.2 The Problem of Partitioning

By their nature, network applications for mobile computing involve cooperation among multiple sites. For these applications, which are characterized by reliability and reconfigurability requirements, possible partitioning of the communication network is an extremely important aspect of the environment. In addition to accidental partitioning caused by failures and node movement, mobile computing systems typically support *disconnected operation* , i.e., a mobile host may intentionally decide to disconnect itself from the network and work only locally, which is an additional cause of partitioning.

Intuitively, partitions correspond to maximal connected components of the logical graph representing the "reachable" relation among processes. As such, they can be defined only in the context of specific communication primitives. For ex-

[4]For every correct process p, if a process q remains unreachable from p, then eventually p will always suspect q.

[5]For every correct process p, if a process q remains reachable from p, then eventually p will no longer suspect q.

[6]Informally, a *self-stabilizing* system is a system that can automatically recover, in a finite number of steps, following the occurrence of transient faults. Once the system returns in a legal configuration, it remains in the legal configuration thereafter, until a subsequent fault occurs. No startup or initialization is necessary because the system stabilizes to the correct behavior by itself [69].

ample, two processes may appear to belong to two different partitions with respect to "ping" messages, but the same two processes may appear in the same partition when communicating through email. This is because the two communication services considered have significantly different message buffering, timeout and retransmission properties.

The nature of partitioning will determine the quality for the application in terms of which of its services are available where, and at what performance levels. In other words, partitioning may result in service *reduction* or service *degradation* but need not necessarily render application services completely unavailable.

Informally, we can define the class of *partition-aware* applications as those that are able to make progress in multiple concurrent partitions without blocking. Service reduction and degradation depend heavily on the application semantics. For certain application classes with strong consistency requirements, it may be the case that all services have to be suspended completely in all but one partition. This situation corresponds to the so-called *primary component* model. For applications with less stringent consistency requirements, partitionable group membership services can provide a useful framework to leverage from.

Babaoglu et al. [71] present three abstract examples of partition-aware applications, which can build upon a partitionable group membership service. These applications are briefly described in the following:

1. *Partitionable Service Activator.* Consider a network service for distributing a continuous stream of data (e.g., audio, video, stock quotes, news headlines) to a collection of subscribers. The data distribution can be provided by any one of a set of *servers* that have access to the data source. The service should be available in every partition that contains at least one server; furthermore, to minimize resource usage, multiple active servers within the same partition should be avoided. New servers may be added and existing ones removed at will by an administrator. The goal is to devise a *service activator* algorithm such that a server can decide when it should be active and when it should be passive. A solution must activate a new server if the current one is removed from the system, if it crashes or if it ends in another partition.

2. *Partitionable Chat.* Consider a service for holding a discussion among a collection of users, e.g., Internet Relay Chat (IRC). Users may contribute to the discussion by *creating* a new thread or by *shouting* messages in an existing thread. Messages are potentially addressed to every user who has joined the discussion. Upon partitioning, the discussion may continue among users in each of the partitions. Shouted messages have to satisfy agreement, integrity, uniqueness and liveness properties of view synchrony messages only within the same partition. No requirements are placed on message threads that span multiple partitions. In other words, upon merging, a user may miss some messages that were shouted in other partitions.

3. *Partitionable Parallel Computation.* Consider a time-intensive computation

such as ray tracing, prime factorization or weather forecasting. The computation can be decomposed into a number of subcomputations that can be carried out independently by a collection of *workers*. New workers may be added and existing ones removed at will. The computation and all relevant input data are known ahead of time to all possible workers. The goal is to conclude the computation in as short time as possible despite crashes, recoveries, partitioning and merges.

Such distributed, partitionable applications are well suited for ad hoc networks due to their peer-to-peer architecture. However, important network applications and services such as web servers, location information databases, and network services (e.g., SNMP) are inherently centralized. These services are often critical to the mobile node's operation such that every node requires constant and guaranteed access to them. When the network partitions, those mobile users that are not in the same partition as the centralized server lose access to the service. To ensure that the service is available to all nodes, a trivial solution is to place the service on every mobile node, so that the service availability is independent of any changes in the network topology. However, this trivial solution incurs a prohibitively high service cost (in terms of the number of servers deployed). Several research works have addressed the problem of maintaining the network-wide coverage of the centralized service in the presence of frequent partitioning, and without incurring high service cost.

Karumanchi et al. [72] assume that there are many designated servers throughout the network. However, the servers are predetermined and fixed, so during network topology changes and network partitioning, their reachability changes. Hence, the work develops run-time heuristics for clients to select servers with the highest likelihood of being accessible, in order to maximize the chances of successful service request.

Hara [73] focuses on data accessibility in ad hoc networks. It assumes that all mobile nodes can store some data replicas. Hence, the work is concerned with the optimal placement of data replicas around the network that achieves high data accessibility in the event of network partitioning, by considering data access frequencies of mobile nodes.

Liang and Haas [74] propose a *virtual service backbone*. The servers are dynamically created and terminated as the network topology changes to ensure network-wide service availability. Further, for ensuring efficiency, only one server is present in a well-connected group of nodes and redundant servers are merged. When servers become inaccessible due to network partitioning, a new server is regenerated. This has two drawbacks. First it relies on the nature of the service being regenerable, which is unlikely for general network services. Without the service being regenerable, the service is lost in the partitioned network. Second, during the period of server failure detection and regeneration, the service is interrupted for the mobile nodes.

Wang and Li [75] utilize observed correlated node mobility patterns [17] to

predict the occurrence of partitioning and take the necessary actions to replicate a server in advance to efficiently provide continuous service availability. Servers know the client velocities as these are piggy-backed on the client requests. Thus, servers can use a sequential clustering algorithm to identify correlated mobility patterns, which are used to predict the time and location of the network partitioning. By calculating the time of service replication, a server can replicate the service onto the partitioned nodes before partitioning occurs. In order to minimize the number of service instances deployed in the network, servers also run a distributed grouping algorithm at regular *service discovery intervals* to discover a set of stable servers. By doing so, the servers in the same stable group monitor each other's presence. As an arbitration, the server with the highest id continues its service, and the others automatically terminate the service instances.

In spontaneously deployed ad hoc networks with no preconfigurations, a mobile node has no prior knowledge about the mobility groups. Moreover, the mobility group membership of a node can change dynamically, as the mobile host may decide to change its course of movement. Therefore, in [75] a distributed grouping algorithm is run by clients to discover mobility group membership based on the stability with respect to distance to neighbor nodes. This algorithm is run at a regular service discovery interval. After each run of the algorithm, a client constructs its stable group and discovers a set of servers. A client selects the best server among the discovered servers; the best server is the one whose relative velocity will allow it to stay in the client group for the longest time (see § 2). If the selected best server is not the client's current server, a service switch occurs. A *reliability counter* is also used by the client for each discovered server as an indicator of the connection stability with that server. At each run of the grouping algorithm, the client adjusts the reliability counter for each server as follows: the counter is incremented if the server is in the client group and halved if the server is not but was discovered previously. A client switches server only if the intended server has the reliability counter higher than a prefixed *switch threshold*.

4.2.3 Group Communication Algorithms for Ad Hoc Networks

This section presents the work on the definition and solution of the group communication services in the context of ad hoc networks. There are currently two approaches to express a stability condition on the system (that is required to make the problem solvable in partitionable networks):

(1) It is assumed that the network becomes connected infinitely often and each time for a sufficient period of time so that at least one pending message can be delivered (i.e., the protocol can perform at least one step). In this case, the liveness property of the specification is given conditionally to such a network stability condition.

(2) The network may partition and never re-merge but it is assumed that mobile nodes and links do not fail.

Some recent work has also specified a location-dependent group membership

service; however, the identification of the fundamental properties that such a service should provide is still an open topic of research. The reason probably may be found in the current lack of real applications for such a service.

Pagani and Rossi [76] present a reliable broadcast protocol based on an underlying multi-cluster multi-hop packet network [77], [78]. The authors presuppose that the network is already structured into clusters. The clustering algorithm uses neighborhood information that a host derives from "I am alive" messages. The clustering algorithm is re-executed in case of topology changes.

The reliable broadcast protocol builds a dynamic forwarding tree involving *cluster heads* (that coordinates the transmission within a cluster and represents the infrastructure to route inter-cluster messages), and *gateways* (hosts that are able to hear from different cluster heads and are used for exchanging messages between two adjacent clusters) in the network. The protocol tolerates communication failures and host mobility. The failure model only allows node and link transient failures that do not cause the loss of state (i.e., a node's state must be saved on stable storage so that it can survive a failure). A liveness property reflects the assumption that in case of temporary disconnections, the network is eventually repaired and remains connected long enough for message and acknowledgment exchange.

The protocol is composed of two phases: in the scattering phase, the message is diffused to all receiver members; in the gathering phase, the acknowledgments are collected from the receivers. A forwarding tree is constructed implicitly during the scattering phase and is destroyed after message stabilization when the protocol terminates.

This protocol can be adapted in a straightforward way to perform multicast by restricting the delivery of message to only the members of the multicast group within each cluster. However, this approach would be wasteful in network bandwidth since the forwarding tree involves all cluster-heads and gateways of the clusters; some or many of these clusters may not belong to the multicast group or may not contain any members of the multicast group. Hence, this algorithm is suitable for only dense multicast and, because of the use of explicit acknowledgment, is not scalable.

Gupta and Srimani [79] propose a reliable multicast protocol and make the following assumptions: (1) there is an underlying reliable unicast routing protocol by which messages can be sent between two non-neighboring nodes; (2) no node leaves or joins the system and, thus, the network graph has always the same node set but different edge sets; (3) transient link failures are handled by the data link layer protocol by using timeouts, retransmission, and per hop acknowledgment. The following network stability property (called "liveness" in the paper) is also required: if there are pending messages for a node u, then eventually the network remains connected long enough so that node u receives at least one of these messages and acknowledges the receipt. Essentially the protocol targets mobile networks with static multicast groups (as explained below) and transient link failures due to node's mobility. Nodes are assumed not to fail (an equivalent assumption is that node failures are transient and do not cause loss of state). A spanning tree is

constructed for each multicast group G. The root of the tree acts as group leader (called core node in the paper), as explained below.

Whenever a node u wants to multicast a message to the members of group G, it sends a MULTICAST message to the leader node of group G, which assigns a sequence number to the message and initiates its dissemination down the multicast tree. It is assumed that node u always knows the identity of the leader node of the destination multicast group (i.e., the multicast groups are static); moreover, the multicast groups are assumed to be open in the sense that any node in the system can multicast a message to any multicast group.

The acknowledgments from the multicast group nodes flow in the reverse direction toward the leader node. Acknowledgment aggregation is used to reduce the bandwidth wastage. A message m stabilizes when the leader node receives acknowledgments from all the multicast nodes. This knowledge about message stabilization is piggy-backed on subsequent multicast messages. A node delivers and deletes its local copy of a message m upon gaining the knowledge about stabilization of m.

A multicast tree may get fragmented due to node movements. One approach to cope with this problem is to reconstruct the multicast tree every time a multicast tree disconnection is detected. In [79] a different approach is pursued by introducing the notion of *forwarding region*, which is used to glue together fragments of the multicast tree. Informally, a forwarding region of a multicast tree node u is the maximal subgraph around u that consists of only non-tree nodes. In order to flood its forwarding region with a message m, node u simply broadcasts m to its neighbors. Any node that receives m forwards it only if the node is not a multicast tree node. The idea is, hence, to flood the message when the multicast tree is fragmented using the forwarding regions so as to restrict the flooding only to regions where the topology has changed.

Huang et al. [80] propose a group membership protocol that tolerates host mobility and frequent disconnections. The protocol relies on location information and employs a conservative notion of logical connectivity that creates the illusion of announced disconnections. Analysis of movement patterns and delays is used to anticipate physical disconnections before they can impact application results.

The mobile ad hoc network is modeled as a *connectivity graph* $C_o = \mathcal{G}(V, E_o)$, where V is the set of mobile hosts and E_o is a set of bi-directional communication links among the hosts. The *logical connectivity graph* $C = \mathcal{G}(V, E)$ is a subgraph of the former, with which it shares the same set of vertexes but may miss some edges. The logical connectivity graph C is used to exclude in advance the subset of links that may fail. The choice of edges to include in C is determined by the group management policy (later described).

Given a node u, the node's group G is the maximum-sized connected subgraph of the logical connectivity graph C that contains node u. The group membership problem is defined as the requirement for each host in the logical connectivity graph to have knowledge of the other members of their group and for such knowledge to be consistent across the entire group at any time. A reliable multicast

service is also provided so that (1) no message can be lost; and (2) a message is delivered at the view in which it was sent. Hosts and links are assumed not to fail. The only threat to maintaining a consistent group membership comes from the host mobility. Motion is assumed to be continuous, random, and subject to a known maximum speed limitation. Hosts may shut down only intentionally by declaring their intention before powering down the transmitter. An underlying protocol providing message delivery with bounded delay between two adjacent nodes is also assumed.

The group membership policy proposed is such that a host is admitted into the group, only if the host can guarantee reliable message delivery from other hosts present in the group. A host transmission range R is restricted to a smaller range r, where the host can provide message delivery guarantee, called *safe transmission zone*. By restricting the range of a host u, it becomes feasible to complete any communication between two hosts u and v before v moves out of the transmission range of u. To determine r, it is assumed that hosts u and v are moving away from each other at a velocity which is the larger between their two velocities, V_{max}, and it takes a time T_{tran} to complete a transaction between the two hosts, i.e., the round trip time between the two hosts is bounded by T_{tran}. The displacement between u and v within time T_{tran} is no more that $2V_{max}T_{tran}$ and, hence, $r = R - 2V_{max}T_{tran}$. When a host u joins a group, a displacement $2V_{max}T_{net}$ is considered, where T_{net} is the round trip time within the group. If u is within the safe zone of any neighbor that is already in the group, then u is admitted to the group.

Partition anticipation is used to preventing mobility-induced unannounced disconnection. All group members periodically send their location information to the group leader, which in turn updates the group map, a data structure recording the group members' last reported locations. Whenever the group map is updated, the leader of the group checks the new configuration to see if the group will soon be in danger of being physically split. In such a case, the leader anticipates the partition by issuing a partition transaction order to the group members.

The algorithm suffers several limitations, as also the authors note. (1) It requires a known maximum node speed; indeed, unbounded speed range is another possible source of unannounced disconnection due to the low speed requirement for most wireless networks (e.g., a GSM or PCS device can communicate only if its speed is lower than about $50m/s$, for a DECT device the speed limit is about $11m/s$). (2) The space is assumed to be free, in the sense that no obstacles can be present in the path between two hosts that are at a distance less that the safe distance. This is reflected in the assumption of continuous node movement. (3) The algorithm relies on an underlying routing protocol that can provide a small and bounded delay bound for group control messages (note that the safe distance is defined in terms of round trip times). (4) Node crash failures are not contemplated.

While in distributed computing a group is usually defined as a named entity to which a host may wish to join, in the context of mobile environments additional aspects of group communication may need to be considered. A group membership

may be not only affected by the state of nodes and links, but also by the location of mobile nodes. For example, a police dispatch service may wish to coordinate the actions of all non-busy troop cars within a kilometer of a crime site. Indeed, there has been work that aims at specifying and solving a location-based group membership problem.

Prakash and Baldoni [81] provide a first attempt to define a location-based group membership service. It is assumed that the network stays connected and there are no failures. Only changes due to mobility are considered. The architecture proposed is composed of a *proximity layer* between the *group membership layer* and the underlying mobile network.

The proximity layer consists of a protocol that uses services of the MAC sublayer. The MAC sublayer provides point-to-point communication and periodic beacons within transmission range d. Two nodes are connected if their distance is less than d. The proximity layer protocol is run periodically and it is assumed that the communication at this level is synchronous and the one-way message communication delay is bounded. For this, collision-free protocols like the bit-map protocol or binary countdown, or limited contention protocols like adaptive tree walk are suggested. The goal of the proximity layer is to find all nodes within distance D from a given node p. It is assumed that during a given *D-proximity* test, the separation between nodes may change due to mobility, but the connectivity graph remains unchanged. Proximity layer messages are *location stamped*. For $D \leq d$, a node p can directly reach all nodes in its *D-proximity* with a single round of messages. For $D > d$, a multi-round synchronous algorithm is executed.

Because of node mobility, the result of the *D-proximity* test can have some inaccuracy and this can lead to a possible violation of the safety requirement of a mobile application. This problem is typical of mobile distributed systems. The authors propose a method to build a margin of safety in a group membership determination by assuming a maximum speed for mobile nodes and, hence, incrementing D with an safety term D' so as to take into account a worst case scenario.

The group membership layer communication, atop of the proximity layer, is modeled as timed asynchronous [64]. This is because communication between nodes participating in group communication is not the same as the point-to-point communication at the proximity layer and, so, messages may have finite but unpredictable delays. The protocol proposed is based on the three round protocol described by [82].

Briesemeister [83] proposes a routing architecture for ad hoc network in the context of inter-vehicle communication. Vehicles are equipped with computer-controlled radio modems allowing them to contact other equipped vehicles in their vicinity. Vehicles are also aware of their location by using, for example, a Global Position System (GPS). By exchanging information, vehicles on a highway build knowledge about the local traffic situation. Once an equipped vehicle slows down significantly, it considers that it is inside a congested area. Then, it starts communicating with equipped vehicles nearby to share information on the driving situation. Vehicles inside the congested area create a dynamic group and try to establish com-

mon knowledge about the size, the beginning and the end of the congestion.

Frequent topology changes, scarce bandwidth, and large-scale coverage prevent the hosts from exchanging position and routing table updates throughout the whole network. Instead, hosts maintain network information only about the local environment. Only direct neighbors that are in the radio communication range of each other exchange position information. Each vehicle inside the group compares its own position with the location of other nearby vehicles. Then, every vehicle decides whether it is at the beginning, in the middle, or at the end of the congested area. Vehicles flood messages to geographic regions or to other vehicles with certain constraints in velocity, relative positions, or similar vehicular parameters. The receivers of the flooded message use their knowledge of the local environment to decide whether they match the intended destination of the message. Moreover, since the inter-vehicle ad hoc network is likely to suffer from partitions, vehicles driving in the opposite direction transport the message backward on the road and close gaps in the network topology.

The problem formulation requires the mobile nodes to aggregate into a dynamic group, which includes all vehicles that slow down in the same driving direction on the highway. Because of the impossibility result of primary-component group membership in asynchronous systems with crash failures [57], the work suggests reducing the group membership service to the local environment of a node. The so-called *Localized Group Membership Service (LGMS)* is formally defined and employed in solving the congested area detection problem.

The LGMS tracks the membership only of adjacent neighbors. Changes in the localized group membership—existent neighbors join or leave the group voluntarily or crash, new members move into vicinity—are installed as local views at each host. Different vehicles' views differ according to the vehicle's neighborhood relation with other vehicles and due to transmission failures.

Although the LGMS proposes interesting solutions to the problem of communication in partitioned ad hoc networks, it is tailored and limited in its scope to the specific problem the author aims to solve, i.e., congestioned area detection on highways. Indeed, there is no support for reliable unicast or multicast communication from any sender to any destination but the information is guaranteed to flow, in case of partitions, only in the opposite direction to the vehicles movement.

Dolev, Schiller and Welch [84] propose a randomized self-stabilizing group membership service for ad hoc networks. The group membership list is carried through the network by the random walks of a mobile agent. This approach to information dissemination contrasts with the use of flooding—that can result in heavy traffic—and the use of a distributed spanning tree (e.g., TORA [27])—that can perform poorly when changes are very frequent. A *nice execution* is defined as a system execution in which there exist a single agent in the system, and the single agent arrives at every nodes in the system at most every M consecutive moves—M is a fixed constant. The papers provides the probability of having a nice execution in three different scenarios. For every nice execution, the proposed membership service satisfies two properties: (1) if a node requests to join the group then it will

be eventually admitted, and (2) if no node requests to join or leave the group then no new view will be generated.

An agent carries the view identifier vid, the list of members, $members_g$, and a list lvs of counters lv_i, each of which is associated to a node $p_i \in members_g$. Whenever an agent visits a node p_i, all $lv_j \in lvs$ are decremented by 1. A process p_j is considered *active member* of the group if and only if $lv_j > 0$. If the receiving node p_i wants to join the group (or simply stay in the group if already present), then p_i is inserted in $members_g$ (in case of join) and lv_i is set to ttl_i, a p_i's predefined constant representing the expected number of agent moves required for the agent to cover the communication graph. After, if p_i discovers that the set of members has changed, a new view identifier is generated. Finally, a p_i's neighbor is randomly chosen for the agent to be sent to.

To make the algorithm self-stabilizing, two techniques are employed. (1) To ensure that there is at least one agent in the system, each node p uses timeouts to detect whether an agent has not visited the node recently enough. In such a case, a new agent is generated, which will contain a group including only node p. (2) To ensure that there is at most one agent in the system, a single agent is generated from the "collision" of two or more agens—agents collide by reaching the same node at the same time.

The paper also proposes a best effort, total-order, group multicast service based on the above membership service. The single agent accumulates the history of the membership views and the messages multicast within each view. Whenever an agents arrives at a node, the node can receive and deliver all messages multicast in the view of which it is a member. Moreover, the node can append its new messages to the agent's history so that they can be received by the other nodes.

4.3 Leader Election and Distributed Mutual Exclusion in Ad Hoc Networks

This section presents the work on the definition and solution of the leader election and distributed mutual exclusion problems in the context of ad hoc networks.

Hatzis et al [85] propose leader election algorithms for mobile ad hoc networks. The algorithms are classified in (1) *Non-Compulsory* protocols, which do not affect the motion of the nodes and try to take advantage of the mobile hosts natural movement by exchanging information whenever mobile hosts meet incidentally; and (2) *Compulsory* protocols, which determined the motion of some or all the nodes according to a specific scheme in order to meet the protocol demands (i.e., meet more often, spread in geographical area, etc.). In both protocol classes, it is assumed that the mobile node moves in a bounded three-dimensional space S, which is quantized by some regular polyhedron, as explained in the following.

A mobile host transmission range is represented by a sphere tr centered at the mobile host. The sphere tr is approximated with a regular polyhedron tc (e.g., a cube) with volume $V(tc)$ less the the volume of the sphere $V(tr)$ and such that if a mobile host inside tc broadcasts a message, this message is received by any

other host in tc. The graph $\mathcal{G}(V, E)$ corresponding to the fixed quantization of S is constructed as follows: a vertex $u \in V$ represents a polyhedron of volume $V(tc)$; an edge (u, v) is in E iff the corresponding polyhedra are adjacent. A host can move anywhere in S but at any instance of time it is inside a specific polyhedron tc and, hence, resides in only one vertex of $\mathcal{G}$. By moving, a host can pass from one polyhedron to an adjacent polyhedron. Note that the number n of vertices in $\mathcal{G}$ approximates the ratio between the volume of the space S, $V(S)$, and the volume of the space occupied by the transmission range of an host, $V(tr)$. In the extreme case where $V(S) \approx V(tr)$ (the transmission range of the hosts approximates the space in which they are moving), then $n = 1$.

In order for these algorithms to work, the mobile nodes should know in advance the type and the dimensions of the polyhedron that is used for the quantization of S; furthermore, the nodes must be able to measure the distance that they cover when they move so that they can determined whether they have covered enough distance to reach a new vertex of $\mathcal{G}$. Also, the *Non-Compulsory* protocols might never elect a unique leader and the *Compulsory* protocols force the nodes to perform a random walk. Neither of the protocol classes addresses the issue of creation of new components due to partitioning and merging of components.

Malpani et al. [86] propose a leader election algorithm based on TORA [27], which is a routing algorithm for mobile ad hoc networks. Each node keeps a value, called height, and links are logically considered to be directed from higher to lower heights. The heights are manipulated on topology changes, so that the logical graph in each connected component eventually forms a leader-oriented DAG, i.e., a DAG in which the leader is the only sink. When a partition from the current leader is detected, a new leader is elected and its id is propagated throughout the component. When two components merge, a contest takes place between the leaders so that the winner's id is propagated and wipes out the loser's id.

The network is modeled as a dynamically changing, not necessarily connected, undirected graph and the following assumptions are made: (1) nodes have unique ids; (2) nonfaulty links provide reliable FIFO-order communication; (3) only one link failure or link formation occurs at a time. The paper also proposes a variation of the algorithm to handle multiple concurrent changes but no correctness proof is given for it. Although not explicitly mentioned by the authors, as the algorithm is based on TORA, it is also assumed that mobile hosts have synchronized local clocks (for example, via GPS).[7] Finally, no simulation or performance evaluation of the algorithm is provided.

Walter et al. [87] propose a token-based algorithm for mutual exclusion in ad hoc networks. The algorithm uses the concept of link height and the partial reversal technique used in [27] to construct a token-oriented DAG of the network, i.e. a DAG in which the node possessing the token is the only sink. The assumptions on the mobile nodes and network are the following: (1) nodes have unique ids; (2) nodes do not fail; (3) nonfaulty links provide FIFO-order reliable commu-

[7] See [27] for an attempt to relax this assumption.

nication; (4) message delays obey the triangle inequality (i.e., messages that travel 1 hop will be received before messages sent at the same time that travel more than 1 hop); (5) the network does not partition. The mutual exclusion problem is so formulated: (*Mutual Exclusion*) at most one node is in the critical section at a given time; (*No Starvation*) once link failures cease, if a node is waiting to enter the critical section, then it will eventually enter the critical section. The hypothesis that link failures cease is required because an adversarial pattern of link failures can cause starvation. The paper provides a correctness proof for the algorithm as well as preliminary simulation results.

Walter et al. [88] extend this work to the k-mutual exclusion problem. In this problem, at most k processes may be in the critical section at any given time. Similarly to [87], the algorithm uses the concept of link height and the partial reversal technique to ensure that all non-token holding processes always have a path to some token holding process.

The assumptions on the mobile nodes and network are the following: (1) nodes have unique ids; (2) nodes can crash as long as not all token holders crash; (3) non-faulty links provide FIFO-order reliable communication; (5) the network can partition but only connected components having at most one token holder can continue running. The algorithm is described and simulation results are provided; however, no token regeneration procedure is considered. In the following, an overview of the algorithm is presented.

A DAG is maintained on the physical wireless links of the ad hoc network. A node's height is represented by a triple of integers. Links are considered to be directed from nodes with higher height to nodes with lower height. A total order on the node heights is ensured by having the last integer in the triple to be the unique node id. A node's height is included in all messages the node sends in the algorithm. Three types of messages are considered: *Request*, *Token*, and *LinkInfo*. The *LinkInfo* messages are sent by a node to its neighbors whenever the node changes its height, so that the neighbors can adapt to the node's height change. The usage of the *Request* and *Token* messages is illustrated in the following.

Initially, the token holders are nodes $0, \ldots, k-1$. When the application process at a node i makes a request for entering the critical section, node i's identifier is enqueued in the node request queue Q_i and the application process is suspended. Two cases are then considered. (1) If the node i does not hold the token and the node's identifier has been enqueued in an empty request queue Q_i, then node i sends a *Request* message to its lowest neighbor j. The algorithm is such that the *Request* messages received at a node j from higher neighbors cause the node j to enqueue the identifiers of these neighbors in Q_j in the order in which their *Request* messages are received by j. (2) If node i holds the token, then it extracts the top element of Q_i. If this element is the node i's own identifier, then the application process at i is resumed and accesses the critical section. Otherwise, node i sends a *Token* message to its neighboring node j whose identifier was just dequeued. When a node j receives a *Token* message, it dequeues the top element of Q_j and either enters the critical section (if its own identifier was the one extracted) or, in turn,

sends a *Token* message to its neighboring node whose identifier was just dequeued.

Non-token holding nodes ensure that they have at least one lower neighbor at all times (recall that messages and, hence, *Request* messages are always sent on outgoing links) by using the partial reversal technique so as to raise the first two integers of its height triple and, hence, create at least one outgoing link.

Token holding nodes ensure that they always have at least one higher neighbor (so that *Request* messages can always be delivered to them) by using the partial reversal technique so as to lower the first two integers of the node's height. In particular, a token recipient may modify the first two integers in its height triple on receiving a *Token* message so that its height is always lower than the height of the sender of the *Token* message.

A node i removes its neighboring node j's identifiers from its request queue Q_i when the link between i and j fails. If a link incident to a node i is reversed, then node i removes all the identifiers from Q_i that are lower then the node i's identifier. If the resulting Q_i is not empty (i.e., it contains the node i's identifier), then node i forwards a *Request* message again to its current lowest neighbor. This is to ensure that the node's requests are not lost as result of the link reversal but always have a chance to re-propagate on a new route toward a token holder.

5 Conclusions: A Fault Tolerance Perspective

Mobile computing is not, strictly speaking, a new computing paradigm (the literature on unicast routing in ad hoc networks is fairly voluminous, for instance); nevertheless, dependable mobile computing probably can be considered as such since, as shown in the previous sections, many issues in system modeling and problem definition are still open. Similarly, the proposed algorithms need strong, and often unrealistic, assumptions on the wireless network's or nodes' behavior.

The logic steps to follow when approaching a new computing model can be delineated as follows [89]: (1) identification of the possible applications so as to foresee as much as possible the expected requirements and the working scenarios; (2) factorization and formalization of these requirements and scenarios in generic problems and system models; and (3) provision of solutions (i.e., algorithms) and their verification. None of these steps can be considered satisfactorily made for dependable mobile computing. A reason may be found in the fact that substantial, realistic applications, with strong dependability requirements, are still to be developed for wireless mobile networks and, hence, it is not clear yet what kind of services or degree of consistency to provide. Besides, the theoretical impossibility result due to the unavoidable network partitions, makes the problems application specific and, hence, difficult to generalize.

General speaking, different applications classes are for different wireless network types, and vice versa; therefore, it is not feasible to move the classical problems of distributed computing to mobile wireless networks as they are. Moreover, mobility and location are characteristic peculiar of mobile networks and should be

considered as well. Location-based routing and location-based membership service are examples of problems specific to mobile networks.

Collaborative (peer-to-peer) applications seem best fit ad hoc networks and may benefit from a group communication paradigm. Similarly, a regenerable server model can fit an ad hoc network as long as server regeneration at each partition is feasible (see [75]), but this heavily depends on the semantic of the services to be provided. In our opinion, client-server (centralized) applications (e.g., databases) do not fit well ad hoc networks and are probably more suited for a fixed infrastructure network with the servers residing on the wired networks, so that clients always know where the servers are. Again, the problem of solving inconsistencies when globally committing changes that have been committed locally is application specific. On the other hand, sensor networks are more for query-based applications where an external user monitors the phenomenon sensed by the network. Here, the asymmetry of the communication and the intrinsic redundancy within a sensor network may suggest solutions to dependability different than those for ad hoc networks.

Work on dependable mobile computing is characterized by the lack of a proper theoretical model. Typically, protocols implicitly assume a synchronous model for the network, i.e., reliable communication in one step (e.g., those in § 4.3), while a timed asynchronous model [64] would be more appropriate. Moreover, the failure model considered is often restricted to mobility (e.g., topology changes) with the assumption that the network stays connected (e.g., [81]). Crash failures are not always considered and we are not aware of any work that considers Byzantine node failures.

Current fault-tolerant algorithms for partitionable mobile ad-hoc networks assume either that (1) mobile nodes move in a free space with bounded physical speed and never fail, or that (2) the network re-connects infinitely often so that the algorithm can make progress. Although in a real system (2) may hold for most of the time (i.e., with high probability), it can still be the case that waiting for network components to re-merge may require unbounded node resources and communication delays. Therefore, we retain that in practice the problem of permanent partitioning cannot be avoided and must be faced together with the presence of obstacles in the space and the possibly unbounded node velocity. In this direction, we individuate two classes of inconsistency in partitionable mobile networks: inconsistency originating from the presence of multiple partitions (*Inter-Partition Inconsistency*); and inconsistency originating within a partition (*Intra-Partition Inconsistency*). These are discussed in the following.

Inter-Partition Inconsistency. This is due to the possibility for the network to partition and never re-merge. Network partitioning can be realistically coped with either (1) by exploiting the nodes' natural movement (e.g., [44]) or, when this does not suffice, (2) by employing dedicated nodes (the so-called *carriers*) to buffer messages yet to be delivered and move in a coordinated way over the space so as

to join in time the network partitions that cannot be joined in space (e.g., [44]). A simple mobility model for these carriers is the random-walk model (as in [46]), but it is only feasible in free space where no obstacles are present. It is necessary an in-depth study of a correlated carriers' movement, possibly by obtaining and exploiting information of obstacles present in the space and of the position and movement of the the different partitions.

Intra-Partition Inconsistency. This is due to the presence of obstacles in the space and to unbounded node velocity, which break the correspondence between the location-based neighboring relation and the logical connectivity relation, on which some work is based (e.g., [80], [81]). Note that, although most wireless network technologies can provide communication only when a mobile device speed is lower than a technology related maximum speed (e.g., a GSM device can communicate only if its speed is lower than about $50m/s$), this does not necessarily imply that a given mobile node will always move at a physical speed lower than such a bound.

When a mobile node u moves too fast (or is hidden by an obstacle), it simply becomes invisible to the network. Once the node u slows down (or surpasses the obstacle), it reappears in a position x' different from the position x in which it disappeared. This *teletransport* effect may be a source of inconsistency for the global state of the system and, to the best of our knowledge, has not been addressed in the literature. Note that the teletransport phenomenon may not be modeled with a crash failure. Indeed, a crashed node that recovers typically does so by restarting from a legal initial state, so that it can be reintegrated in the network by performing a precise reintegration protocol. On the other hand, it is not difficult to think of scenarios in which, because of the teletransport effect, when the mobile node u reappears in the network, the node u and the other network nodes have an inconsistent view of the situation, i.e., the global system is in an illegal configuration. This is similar to having the mobile node u to be restarted at x' in an arbitrary state, and so can be dealt with self-stabilizing algorithms.

References

[1] G. H. Forman and J. Zahorjan, "The challenges of mobile computing," Tech. Rep. TR-93-11-03, University of Washington, 1993.

[2] C.-K. Toh, *Ad Hoc Mobile Wireless Networks.* Prentice Hall PTR, 2001.

[3] K. Chandran et al., "A feedback based scheme for improving tcp performance in ad hoc wireless networks," in *Proc. Int'l Conf. on Distributed Systems (ICDS)*, 1998.

[4] D. Kim, C.-K. Toh, and Y. Choi, "TCP BuS: Improving TCP performance in wireless ad hoc networks," *Journal of Communications and Networks*, vol. 3, no. 2, 2001.

[5] J. L. et al., *Dependability Handbook*. Laboratory for Dependability Engineering, preliminary version ed., 1998.

[6] F. Stajano, *Security for ubiquitous computing*. Wiley series in communications networking and distributed systems., J. Wiley, 2002.

[7] G. Chen and D. Kotz, "A survey of context-aware mobile computing research," Tech. Rep. TR2000-381, Dept. of Computer Science, Dartmouth College, 2000.

[8] I. Akyildiz, W. Su, Y. Sankarasubramaniam, and E. Cayirci, "Wireless sensor networks: A survey," *Computer Networks Journal*, vol. 38, no. 4, pp. 393–422, 2002.

[9] J. Hill, R. Szewczyk, A. Woo, S. Hollar, D. E. Culler, and K. S. J. Pister, "System architecture directions for networked sensors," in *Proc. 9th Int'l Conf. on Architectural Support for Programming Languages and Operating Systems*, pp. 93–104, 2000.

[10] L. Schwiebert, S. K. S. Gupta, and J. Weinmann, "Research challenges in wireless networks of biomedical sensors," in *Proc. ACM/IEEE Conf. on Mobile Computing and Networking (MOBICOM)*, pp. 151–165, 2001.

[11] A. McDonald and T. Znati, "A mobility-based framework for adaptive clustering in wireless ad hoc networks," *IEEE Journal on Selected Areas in Communications*, vol. 17, no. 8, pp. 1466–1486, 1999.

[12] S. Jiang, D. He, and J. Rao, "A prediction-based link availability estimation for mobile ad hoc networks," in *Proc. IEEE INFOCOM*, pp. 1745–1752, 2001.

[13] W. Su, S.-J. Lee, and M. Gerla, "Mobility prediction and routing in ad hoc wireless networks," *Int'l Journal of Network Management*, 2000.

[14] D. Tang and M. Baker, "Analysis of a local-area wireless network," in *Proc. ACM/IEEE Int'l Conf. on Mobile Computing and Networking (MOBICOM)*, pp. 1–10, 2000.

[15] X. Hong, M. Gerla, G. Pei, and C. Chiang, "A group mobility model for ad hoc wireless networks," in *Proc. ACM/IEEE MSWiM*, 1999.

[16] O. Johansson, T. Larsson, N. Hedman, B. Mielczarek, and M. Degermark, "Scenario-based performance analysis of routing protocols for mobile ad hoc networks," in *Proc. ACM/IEEE Int'l Conf. on Mobile Computing and Networking (MOBICOM)*, pp. 195–205, 1999.

[17] K. H. Wang and B. Li, "Group mobility and partition prediction in wireless ad hoc networks," in *Proc. IEEE Int'l Conf. on Communications (ICC)*, 2002.

[18] T. Camp, J. Boleng, and V. Davies, "Mobility models for ad hoc network research," *Wireless Communications & Mobile Computing (WCMC): Special issue on Mobile Ad Hoc Networking: Research, Trends and Applications*, 2002.

[19] D. B. Johnson and D. A. Maltz, "Dynamic source routing in ad hoc wireless networks," in *Mobile Computing* (Imielinski and Korth, eds.), pp. 153–181, Kluwer Academic Publishers, 1996.

[20] E. Royer and C. Toh, "A review of current routing protocols for ad-hoc mobile wireless networks," *IEEE Personal Communications*, pp. 46–55, 1999.

[21] A. Kashyap, H. Nishar, and P. Agarwal, "Survey on unicast routing in mobile ad hoc networks.".

[22] J. Broch, D. A. Maltz, D. B. Johnson, Y.-C. Hu, and J. Jetcheva, "A performance comparison of multi-hop wireless ad hoc network routing protocols," in *Proc. 4th ACM/IEEE Int'l Conf. on Mobile Computing and Networking (MOBICOM)*, pp. 85–97, 1998.

[23] C. Perkins and P. Bhagwat, "Highly dynamic destination-sequenced distance-vector routing (DSDV) for mobile computers," in *ACM Conf. on Communications Architectures, Protocols and Applications (SIGCOMM)*, pp. 234–244, 1994.

[24] L. R. F. Jr. and D. R. Fulkerson, *Flows in Networks*. Princeton University Press, 1962.

[25] C. Chiang, H. Wu, W. Liu, and M. Gerla, "Routing in clustered multihop, mobile wireless networks," in *Proc. IEEE Singapore Int'l Conf. on Networks (SICON)*, pp. 197–211, 1997.

[26] C. E. Perkins and E. M. Royer, "Ad-hoc on-demand distance vector routing," in *Proc. 2nd IEEE Wksp. Mobile Computer Systems and Applications*, pp. 90–100, 1999.

[27] V. D. Park and M. S. Corson, "A highly adaptive distributed routing algorithm for mobile wireless networks," in *Proc. IEEE INFOCOM*, pp. 1405–1413, 1997.

[28] C.-K. Toh, "A novel distributed routing protocol to support ad-hoc mobile computing," in *Proc. IEEE 15th Int'l Phoenix Conf. on Computers and Communications*, pp. 480–486, 1996.

[29] R. Dube, C. Rais, K. Wang, and S. Tripathi, "Signal stability based adaptive routing (SSA) for ad hoc mobile networks," *IEEE Personal Communication Magazine*, vol. 4, no. 1, pp. 36–45, 1997.

[30] Z. J. Haas and M. Perlman, "The performance of query control schemes for the zone routing protocol," *IEEE/ACM Trans. on Networking*, vol. 9, no. 4, pp. 427–438, 2001.

[31] J. Hightower and G. Borriello, "Location systems for ubiquitous computing," *IEEE Computer*, vol. 34, pp. 57–66, August 2001.

[32] M. Mauve, J. Widmer, and H. Hartenstein, ""a survey on position-based routing in mobile ad hoc networks," *IEEE Network Magazine*, vol. 15, no. 6, pp. 30–39, 2001.

[33] Y.-B. Ko and N. H. Vaidya, "Location-aided routing (LAR) in mobile ad hoc networks," *ACM/Baltzer Wireless Networks (WINET) journal*, vol. 6, no. 4, pp. 307–321, 2000.

[34] L. Blazevic, L. Buttyan, S. Capkun, S. Giordano, J.-P. Hubaux, and J.-Y. L. Boudec, "Self-organization in mobile ad hoc networks: The approach of Terminodes," *IEEE Communications Magazine*, vol. 39, no. 6, 2001.

[35] S. Capkun, M. Hamdi, and J. Hubaux, "GPS-free positioning in mobile ad-hoc networks," in *Proc. 34th Hawaii Int'l Conf. On System Sciences (HICSS)*, 2001.

[36] M. Silventoinen and T. Rantalainen, "Mobile station emergency locating in GSM," in *Proc. IEEE Int'l Conf. on Personal Wireless Communications*, 1996.

[37] P. Lettieri and M. Srivastava, "Advances in wireless terminals," *IEEE Personal Communications*, vol. 6, no. 1, pp. 6–18, 1999.

[38] K. M. Sivalingam, J.-C. Chen, P. Agrawal, and M. Srivastava, "Design and analysis of lowpower access protocols for wireless and mobile ATM networks," *ACM/Baltzer Wireless Networks*, vol. 6, no. 1, pp. 73–87, 2000.

[39] J.-H. Chang and L. Tassiulas, "Energy conserving routing in wireless ad-hoc networks," in *Proc. IEEE INFOCOM*, pp. 22–31, 2000.

[40] R. Ramanathan and R. Hain, "Topology control of multihop wireless networks using transmit power adjustment," in *Proc. IEEE INFOCOM*, pp. 404–413, 2000.

[41] J. R. Lorch and A. J. Smith, "Software strategies for portable computer energy management," *IEEE Personal Communications Magazine*, vol. 5, no. 3, pp. 60–73, 1998.

[42] C. E. Jones, K. M. Sivalingam, P. Agrawal, and J.-C. Chen, "A survey of energy efficient network protocols for wireless networks," *Wireless Networks*, vol. 7, no. 4, pp. 343–358, 2001.

[43] Q. Li and D. Rus, "Sending messages to mobile users in disconnected ad-hoc wireless networks," in *Proc. ACM/IEEE Int'l Conf. on Mobile Computing and Networking (MOBICOM)*, pp. 44–55, 2000.

[44] A. Vahdat and D. Becker, "Epidemic routing for partially connected ad hoc networks," Tech. Rep. CS-200006, Department of Computer Science, Duke University, Durham, NC, 2000.

[45] I. Chatzigiannakis, S. Nikoletseas, and P. Spirakis, "Analysis and experimental evaluation of an innovative and efficient routing protocol for ad-hoc mobile networks," *Lecture Notes in Computer Science*, vol. 1982, pp. 99–111, 2001.

[46] I. Chatzigiannakis, S. Nikoletseas, N. Paspallis, P. Spirakis, and C. Zaroliagis, "An experimental study of basic communication protocols in ad-hoc mobile networks," *Lecture Notes in Computer Science*, vol. 2141, pp. 159–169, 2001.

[47] K. A. Amin and A. R. Mikler, "Dynamic agent population in agent-based distance vector routing," in *Proc. 2nd Int'l Wksp. on Intelligent Systems Design and Applications (ISDA)*, 2002.

[48] B. Williams and T. Camp, "Comparison of broadcasting techniques for mobile ad hoc networks," in *Proc. ACM Int'l Symp. on Mobile Ad Hoc Networking and Computing (MOBIHOC)*, pp. 194–205, 2002.

[49] H. Zhou and S. Singh, "Content based multicast (CBM) in ad hoc networks," in *Proc. 1st Annual Wksp. on Mobile Ad Hoc Networking and Computing*, pp. 51–60, 2000.

[50] X. Jiang and T. Camp, "Review of geocasting protocols for a mobile ad hoc network," in *Proc. Grace Hopper Celebration (GHC)*, 2002.

[51] E. Bertino, E. Pagani, and G. P. Rossi, *Recovery in Database Management Systems*, ch. Fault Tolerance and Recovery in Mobile Computing Systems. Prentice Hall, 1998.

[52] D. Barbara, "Mobile computing and databases–a survey," *Knowledge and Data Engineering*, vol. 11, no. 1, pp. 108–117, 1999.

[53] E. Pitoura and B. K. Bhargava, "Data consistency in intermittently connected distributed systems," *Knowledge and Data Engineering*, vol. 11, no. 6, pp. 896–915, 1999.

[54] C. Bobineau, P. Pucheral, and M. Abdallah, "A unilateral commit protocol for mobile and disconnected computing," in *Proc. 12th Int'l Conf. on Parallel and Distributed Computing Systems (PDCS)*, 2000.

[55] G. Chockler, I. Keidar, and R. Vitenberg, "Group communication specifications: a comprehensive study," *ACM Computing Surveys*, vol. 33, no. 4, pp. 427–469, 2001.

[56] M. J. Fischer, N. A. Lynch, and M. S. Paterson, "Impossibility of distributed consensus with one faulty process," *Journal of the ACM*, vol. 32, pp. 374–382, April 1985.

[57] T. D. Chandra, V. Hadzilacos, S. Toueg, and B. Charron-Bost, "On the impossibility of group membership," in *Proc. 15th Annual ACM Symp. on Principles of Distributed Computing (PODC)*, (New York, USA), pp. 322–330, ACM, 1996.

[58] E. Anceaume, B. Charron-Bost, P. Minet, and S. Toueg, "On the formal specification of group membership services," Tech. Rep. TR95-1534, Computer Science Department, Cornell University, Ithaca, New York 13853, 25, 1995.

[59] A. Ricciardi and K. P. Birman, "Using process groups to implement failure detection in asynchronous environments," in *Proceedings of the Tenth ACM Symposium on Principles of Distributed Computing*, pp. 341–352, 1991.

[60] D. Dolev, D. Malki, and R. Strong, "An asynchronous membership protocol that tolerates partitions," Tech. Rep. CS94-6, Institute of Computer Science, The Hebrew University of Jerusalem, 1994.

[61] R. Friedman and R. V. Renesse, "Strong and weak virtual synchrony in horus," Tech. Rep. TR95-1537, Department of Computer Science, Cornell University, Ithaca, N.Y., 1995.

[62] T. Anker et al., "Scalable group membership services for novel applications," in *Proc. DIMACS Wksp. on Networks in Distributed Computing*, pp. 23–42, 1998.

[63] F. Cristian, "Group, majority, and strict agreement in timed asynchronous distributed systems," in *Proc. 26th Int'l Symp. on Fault-Tolerant Computing (FTCS)*, pp. 178–189, 1996.

[64] F. Cristian and C. Fetzer, "The timed asynchronous distributed system model," *IEEE Trans. on Parallel and Distributed Systems*, vol. 10, no. 6, pp. 642–657, 1999.

[65] A. Fekete, N. Lynch, and A. Shvartsman, "Specifying and using a partitionable group communication service," *ACM Transactions on Computer Systems*, vol. 19, no. 2, pp. 171–216, 2001.

[66] R. Khazan, A. Feke, and N. Lynch, "Multicast group communication as a base for a load-balancing replicated data service," in *Proc. Int'l Symp. on Distributed Computing (ISDC)*, pp. 258–272, 1998.

[67] O. Babaoglu, R. Davoli, and A. Montresor, "Group communication in partitionable systems: Specification and algorithms," *IEEE Trans. on Software Engineering*, vol. 27, no. 4, pp. 308–336, 2001.

[68] A. Montresor, "The Jgroup reliable distributed object model," in *Proc. 2nd Int'l Working Conference on Distributed Applications and Systems*, 1999.

[69] S. Dolev, *Self-Stabilization*. MIT Press, 2000.

[70] S. Dolev and E. Schiller, "Communication adaptive self-stabilizing group membership service," in *Proc. 5th Wksp. on Self-Stabilization (WSS)*, pp. 81–97, 2001.

[71] O. Babaoglu, R. Davoli, A. Montresor, and R. Segala, "System support for partition-aware network applications," in *Proc. 18th Int'l Conf. on Distributed Computing Systems (ICDCS)*, pp. 184–191, 1998.

[72] G. Karumanchi, S. Muralidharan, and R. Prakash, "Information dissemination in partitionable mobile ad hoc networks," in *Proc. IEEE Symp. on Reliable Distributed Systems (SRDS)*, pp. 4–13, 2000.

[73] T. Hara, "Effective replica allocation in ad hoc networks for improving data accessibility," in *Proc. IEEE INFOCOM*, pp. 1568–1576, 2001.

[74] B. Liang and Z. J. Haas, "Virtual backbone generation and maintenance in ad hoc network mobility management," in *Proc. IEEE INFOCOM*, pp. 1293–1302, 2000.

[75] K. H. Wang and B. Li, "Efficient and guaranteed service coverage in partitionable mobile ad hoc networks," in *Proc. IEEE INFOCOM*, 2002.

[76] E. Pagani and G. P. Rossi, "Reliable broadcast in mobile multihop packet networks," in *Proc. Int'l Conf. on Mobile Computing and Networking (MOBICOM)*, pp. 34–42, 1997.

[77] A. Ephremides, J. E. Wieselthier, and J. D. Baker, "A design concept for reliable mobile networks with frequency hopping signaling," in *Proc. IEEE*, pp. 56–73, 1987.

[78] M. Gerla and J. T. Tsai, "Multicluster, mobile, multimedia radio network," *ACM Journal on Wireless Networks*, vol. 1, no. 3, pp. 255–265, 1995.

[79] S. K. Gupta and P. Srimani, "An adaptive protocol for reliable multicast in mobile multi-hop radio networks," in *Proc. 2nd IEEE Wksp. on Mobile Computer Systems and Applications*, pp. 111–122, 1999.

[80] Q. Huang, C. Julien, G. Roman, and A. Hazemi, "Relying on safe distance to ensure consistent group membership in ad hoc networks," in *Proc. 23rd Int'l. Conf. in Software Engineering (ISCE)*, 2001.

[81] R. Prakash and R. Baldoni, "Architecture for group communication in mobile systems," in *Proc. IEEE Symp. on Reliable Distributed Systems (SRDS)*, 1998.

[82] F. Cristian and F. Schmuck, "Agreeing on processor group membership in timed asynchronous distributed systems," Tech. Rep. CSE95-428, University of California, San Diego, 1995.

[83] L. Briesemeister, *Group Membership and Communication in Highly Mobile Ad Hoc Networks*. PhD thesis, School of Electrical Engineering and Computer Science, Technical University of Berlin, Germany, 2001.

[84] S. Dolev, E. Schiller, and J. Welch, "Random walk for self-stabilizing group communication in ad-hoc networks," in *Proc. 21st Simp. on Reliable Distributed Systems (SRDS)*, 2002.

[85] K. P. Hatzis, G. P. Pentaris, P. G. Spirakis, V. T. Tampakas, and R. B. Tan, "Fundamental control algorithms in mobile networks," in *ACM Symposium on Parallel Algorithms and Architectures*, pp. 251–260, 1999.

[86] N. Malpani, J. L. Welch, , and N. H. Vaidya, "Leader election algorithms for mobile ad hoc networks," in *Proc. 4th Int'l Wksp. on Discreete Algorithms and Methods for Mobile Computing and Communications*, pp. 96–103, 2000.

[87] J. E. Walter, J. L. Welch, and N. H. Vaidya, "A mutual exclusion algorithm for ad hoc mobile networks," *ACM abd Baltzer Wireless Networks journal*, vol. 7, no. 6, pp. 585–600, 2001.

[88] J. Walter, G. Cao, , and M. Mohanty, "A k-mutual exclusion algorithm for ad hoc wireless networks," in *Proc. 1st Wksp. on Principles of Mobile Computing (POMC)*, 2001.

[89] X. Défago, "Distributed computing on the move: From mobile computing to cooperative robotics and nanorobotics," in *Proc. 1st Wksp. on Principles of Mobile Computing (POMC)*, 2001.