ADHOC AND SENSOR NETWORKS

R18A1210

DIGITAL NOTES

B.TECH IV YEAR – II SEM

DEPARTMENT OF INFORMATION TECHNOLOGY

MALLA REDDY COLLEGE OF ENGINEERING & TECHNOLOGY

(Autonomous Institution – UGC, Govt. of India)

Recognized under 2(f) and 12 (B) of UGC ACT 1956

(Affiliated to JNTUH, Hyderabad, Approved by AICTE - Accredited by NBA & NAAC – 'A' Grade - ISO 9001:2015 Certified)

Maisammaguda, Dhulapally (Post Via. Hakimpet), Secunderabad – 500100, Telangana State, India

MALLA REDDY COLLEGE OF ENGINEERING AND TECHNOLOGY

IV YEAR B.TECH IT –II SEM **L T/P/D C**

3 -/- / - 3

(R18A1210) ADHOC AND SENSOR NETWORKS (PROFESSIONAL ELECTIVE 4)

OBJECTIVES:

- To understand the concepts of sensor networks.
- To know the MAC and transport protocols for ad hoc networks.
- To learn the security of sensor networks.
- To gain knowledge of the applications of ad-hoc and sensor networks.
- To understand various security practices and protocols of ad-hoc and sensor networks.

UNIT – I

Introduction to Ad Hoc Wireless Networks: Characteristics of MANETS, Applications of MANETS, Challenges Routing In MANETS: Topology based versus position based approaches, Topology based routing protocols, and position based routing, other routing protocols

UNIT – II

Data Transmission In MANETS: The broadcast storm, Multicasting, Geo casting. TCP Over Ad Hoc Networks: TCP protocol overview, TCP and MANETS, Solutions for TCP over Ad Hoc

UNIT – III

Basics Of Wireless Sensors And Applications: The Mica Mote, Sensing and Communication Range, Design Issues, Energy Consumption, Clustering of Sensors, Applications. Data Retrieval In Sensor Networks: Classification of WSNs, MAC Layer, Routing Layer, High- Level Application Layer Support, Adapting to the Inherent Dynamic Nature of WSNs.

UNIT – IV

Security: Security in Ad Hoc Wireless Networks, Key Management, Secure Routing, Cooperation in MANETs, Intrusion Detection Systems Sensor Network Platforms and Tools:

Sensor network Hardware, Sensor Network Programming Challenges, and Node-Level Software Platforms.

UNIT – V

Operating System-Tiny OS: Imperative Language: nesC, Data flow style language: TinyGALS, Node- Level Simulators, NS-2 and its sensors network extension, TOSSIM.

TEXT BOOKS:

1. Ad Hoc and Sensor Networks: Theory and Applications, Carlos de Morais Cordeiro and Dharma Prakash Agrawal, World Scientific Publications / CambridgeUniversityPress,2006.
2. Wireless Sensor Networks: An Information Processing Approach, Feng Zhao, Leonidas Guibas, Elsevier Science Imprint, Morgan Kauffman Publishers,2005.

REFERENCES:

1. Ad Hoc Wireless Networks: Architectures and Protocols, C. Siva Ram Murthy and B. S. Manoj, Pearson Education,2004.
2. Guide to Wireless Ad Hoc Networks, Sudip Misra, Isaac Woungang, and Subhas Chandra Misra, Springer International Edition,2011.
3. Guide to Wireless Sensor Networks, Sudip Misra, Isaac Woungang, and Subhas Chandra Misra, Springer International Edition,2012.
4. Wireless Mesh Networking, Thomas Krag and Sebastin Buettrich, O‘ReillyPublishers,2007.
5. Wireless Sensor Networks – Principles and Practice, Fei Hu, Xiaojun Cao, An Auerbach book, CRC Press, Taylor & Francis Group,2010.
6. Wireless Ad hoc Mobile Wireless Networks-Principles, Protocols and Applications, Subir Kumar Sarkar, et al., Auerbach Publications, Taylor & Francis Group,2008.
7. Wireless Ad hoc Networking, Shih-Lin Wu, Yu-Chee Tseng, Auerbach Publications, Taylor & Francis Group,2007
8. Wireless Ad hoc and Sensor Networks–Protocols, Performance and Control, Jagannathan Sarangapani, CRC Press, Taylor & Francis Group, 2007,rp2010.
9. Security in Ad hoc and Sensor Networks, Raheem Beyah, et al., WorldScientific Publications /Cambridge University Press,2010

COURSE OUTCOMES:

The students should be able to:

- Ability to understand the state-of-the-art research in the emerging subject of Ad Hoc and Wireless Sensor Networks.
- Ability to analyze protocols developed for Ad Hoc and sensor networks.
- Ability to identify and address the security threats in Ad Hoc and sensor networks.
- Ability to solve the issues in real-time application development based on ASN.
- Ability to conduct further research in the domain of ASN.

INDEX

UNIT- I

1. INTRODUCTION

Over recent years, the market for wireless communications has enjoyed an unprecedented growth. Wireless technology is capable of reaching virtually every location on the surface of the earth. Hundreds of millions of people exchange information every day using pagers, cellular telephones, laptops, various types of personal digital assistants (PDAs) and other wireless communication products. With tremendous success of wireless voice and messaging services, it is hardly surprising that wireless communication is beginning to be applied to the realm of personal and business computing. No longer bound by the harnesses of wired networks, people will be able to access and share information on a global scale nearly anywhere thinks about.

Simply stating, a Mobile Ad hoc NETwork (MANET) is one that comes together as needed, not necessarily with any support from the existing Internet infrastructure or any other kind of fixed stations. We can formalize this statement by defining an ad hoc network as an autonomous system of mobile hosts (also serving as routers) connected by wireless links, the union of which forms a communication network modeled in the form of an arbitrary graph. This is in contrast to the well-known single hop cellular network model that supports the needs of wireless communication by installing base stations as access points. In these cellular networks, communications between two mobile nodes completely rely on the wired backbone and the fixed base stations. In a MANET, no such infrastructure exists and the network topology may dynamically change in an unpredictable manner since nodes are free to move. As for the mode of operation, ad hoc networks are basically peer-to-peer multi-hop mobile wireless networks where information packets are transmitted in a store-and-forward manner from a source to an arbitrary destination, via intermediate nodes as shown in Figure 1. As the nodes move, the resulting change in network topology must be made known to the other nodes so that outdated topology information can be updated or removed. For example, as MH2 in Figure 1 changes its point of attachment from MH3 to MH4 other nodes part of the network should use this new route to forward packets to MH2.

Note that in Figure 1, and throughout this text, we assume that it is not possible to have all nodes within range of each other. In case all nodes are close-by within radio range, there are no routing issues to be addressed. In real situations, the power needed to obtain complete connectivity may be, at least, infeasible, not to mention issues such as battery life. Therefore, we are interested in scenarios where only few nodes are within radio range of each other.

Figure 1 raises another issue of symmetric (bi-directional) and asymmetric (unidirectional) links. As we shall see later on, some of the protocols we discuss consider symmetric links with associative radio range, i.e., if (in Figure 1) MH1 is within radio range of MH3, then MH3 is also within radio range of MH1. This is to say that the communication links are symmetric. Although this assumption is not always valid, it is usually made because routing in asymmetric networks is a relatively hard task [Prakash 1999]. In certain cases, it is possible to find routes that could avoid asymmetric links, since it is quite likely that these links imminently fail. Unless stated otherwise, throughout this text we consider symmetric links, with all nodes having identical capabilities and responsibilities.

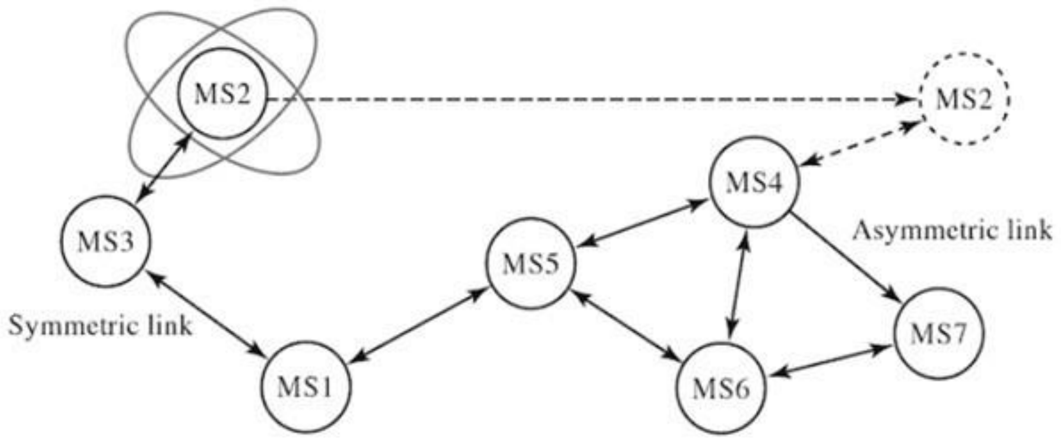


Figure 1.1 - A mobile ad hoc network (MANET)

The issue of symmetric and asymmetric links is one among the several challenges encountered in a MANET. Another important issue is that different nodes often have different mobility patterns. Some nodes are highly mobile, while others are primarily stationary. It is difficult to predict a node's movement and pattern of movement. Table 1 summarizes some of the main characteristics [Duggirala 2000] and challenges faced in a MANET.

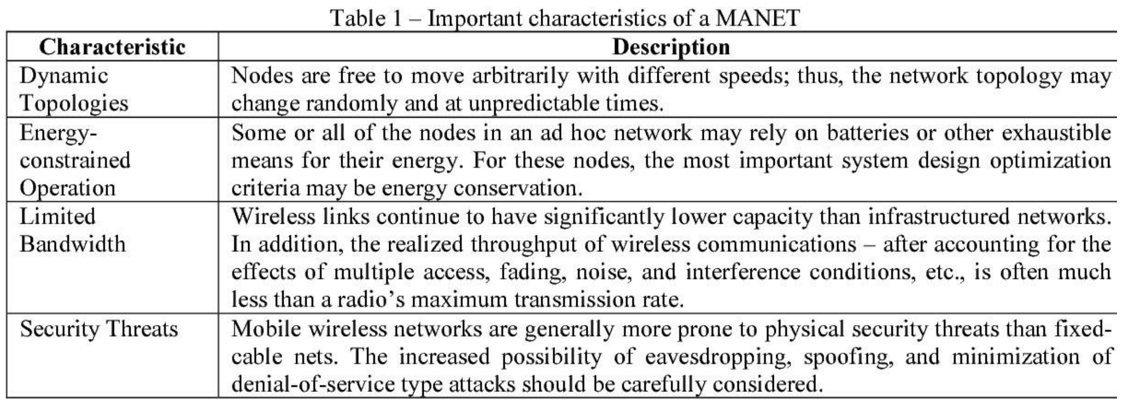


Table 1 – Important characteristics of a MANET

Characteristic	Description
Dynamic Topologies	Nodes are free to move arbitrarily with different speeds; thus, the network topology may change randomly and at unpredictable times.
Energy-constrained Operation	Some or all of the nodes in an ad hoc network may rely on batteries or other exhaustible means for their energy. For these nodes, the most important system design optimization criteria may be energy conservation.
Limited Bandwidth	Wireless links continue to have significantly lower capacity than infrastructured networks. In addition, the realized throughput of wireless communications – after accounting for the effects of multiple access, fading, noise, and interference conditions, etc., is often much less than a radio's maximum transmission rate.
Security Threats	Mobile wireless networks are generally more prone to physical security threats than fixed-cable nets. The increased possibility of eavesdropping, spoofing, and minimization of denial-of-service type attacks should be carefully considered.

Wireless Sensor Networks [Estrin 1999, Kahn 1999] is an emerging application area for ad hoc networks which has been receiving a large attention. The idea is that a collection of cheap to manufacture, stationary, tiny sensors would be able to sense, coordinate activities and transmit some physical characteristics about the surrounding environment to an associated base station. Once placed in a given environment, these sensors remain stationary. Furthermore, it is expected that power will be a major driving issue behind protocols tailored to these networks, since the lifetime of the battery usually defines the sensor's lifetime. One of the most cited examples is the battlefield surveillance of enemy's territory wherein a large number of sensors are dropped from an airplane so that activities on the ground could be detected and communicated. Other potential commercial fields include machinery prognosis, bio sensing and environmental monitoring.

This rest of this text is organized as follows. We initially provide necessary background on ad hoc networking by illustrating its diverse applications. Next, we cover the routing aspect in a MANET, considering both unicast and multicast communication. MAC issues related to a MANET are then illustrated. Following, sensor networks, its diverse applications, and associated routing protocols are discussed. Finally, we conclude this text by discussing the current standard activities at both IETF and the Bluetooth SIG, and also bringing up some open problems that have not received much attention so far and still need to be addressed.

APPLICATIONS OF MANETS

There are many applications to ad hoc networks. As a matter of fact, any day-to-day application such as electronic email and file transfer can be considered to be easily deployable within an ad hoc network environment. Web services are also possible in case any node in the network can serve as a gateway to the outside world. In this discussion, we need not emphasize the wide range of military applications possible with ad hoc networks. Not to mention, the technology was initially developed keeping in mind the military applications, such as battlefield in an unknown territory where an infrastructured network is almost impossible to have or maintain. In such situations, the ad hoc networks having self-organizing capability can be effectively used where other technologies either fail or cannot be deployed effectively. Advanced features of wireless mobile systems, including data rates compatible with multimedia applications, global roaming capability, and coordination with other network structures, are enabling new applications. Some well-known ad hoc network applications are:

• Collaborative Work – For some business environments, the need for collaborative computing might be more important outside office environments than inside. After all, it is often the case where people do need to have outside meetings to cooperate and exchange information on a given project.

• Crisis-management Applications – These arise, for example, as a result of natural disasters where the entire communications infrastructure is in disarray. Restoring communications quickly is essential. By using ad hoc networks, an infrastructure could be set up in hours instead of days/weeks required for wire-line communications.

• Personal Area Networking and Bluetooth – A personal area network (PAN) is a short-range, localized network where nodes are usually associated with a given person. These nodes could be attached to someone's pulse watch, belt, and so on.

CHALLENGES ROUTING IN MANETS:

It has become clear that routing in a MANET is intrinsically different from traditional routing found on infra structured networks. Routing in a MANET depends on many factors including topology, selection of routers, and initiation of request, and specific underlying characteristic that could serve as a heuristic in finding the path quickly and efficiently. The low resource availability in these networks demands efficient utilization and hence the motivation for optimal routing in ad hoc networks. Also, the highly dynamic nature of these networks imposes severe restrictions on routing protocols specifically designed for them, thus motivating the study of protocols which aim at achieving routing stability.

One of the major challenges in designing a routing protocol [Jubin 1987] for ad hoc networks stems from the fact that, on one hand, a node needs to know at least the reachability information to its neighbour's for determining a packet route and, on the other hand, the network topology can change quite often in an ad hoc network. Furthermore, as the number of network nodes can be large, finding route to the destinations also requires large and frequent exchange of routing control information among the nodes. Thus, the amount of update traffic can be quite high, and it is even higher when high mobility nodes are present. High mobility nodes can impact route maintenance overhead of routing protocols in such a way that no bandwidth might remain leftover for the transmission of data packets [Corson 1996].

TOPOLOGY-BASED VERSUS POSITION-BASED APPROACHES:

Routing over ad hoc networks can be broadly classified as *topology based* or *position-based* approaches. Topology-based routing protocols depend on the information about existing links in the network and utilize them to carry out the task of packet forwarding. They can be further
Subdivided as being Proactive (or table-driven), Reactive (or on demand), or Hybrid protocols. Proactive algorithms employ classical routing strategies such as distance-vector or link-state routing and any changes in the link connections are updated periodically throughout the network. They mandate that MHs in a MANET should keep track of routes to all possible destinations so that when a packet needs to be forwarded, the known route can be used immediately. Proactive protocols have the advantage that a node experiences minimal delay whenever a route is needed as a route is immediately obtained from the routing table.

However, proactive protocols may not always be appropriate in MANETs with high mobility. This may cause continuous use of a substantial fraction of the network capacity so that the routing information could be kept current. In addition, the quality of channels may change with time due to the shadowing and fast fading and may not be good to use even if there is no mobility [Lin2005]. On the other hand, reactive protocols employ a lazy approach whereby nodes only discover routes to destinations on-demand. In other words, reactive protocols adopt the opposite approach as compared to proactive schemes by finding a route to a destination only when needed.

Reactive protocols often consume much less bandwidth than proactive protocols, but the delay in determining a route can be substantially large. Another disadvantage is that in reactive protocols, even though route maintenance is limited to routes currently in use, it may still generate a significant amount of network control traffic when the topology of the network changes frequently. Lastly, packets en route to the destination are likely to be lost if the route in use changes. Hybrid protocols combine local proactive and global reactive routing in order to achieve a higher level of efficiency and scalability. For example, a proactive scheme may be used for close by MHs only, while routes to distant nodes are found using reactive mode. Usually, but not always, hybrid protocols may be associated with some sort of hierarchy which can either be based on the neighbors of a node or on logical partitions of the network. The major limitation of hybrid schemes combining both strategies, is that it still needs to maintain at least those paths that are currently in use. This limits the amount of topological changes that can be tolerated within a given time span.

Finally, position-based routing algorithms overcome some of the limitations of topology-based routing by relying on the availability of additional knowledge. These position-based protocols require that the physical location information of the nodes be known. Typically, each or some of the MHs determine their own position through the use of the Global Positioning System (GPS) or some other type of positioning technique

[Hightower2001]. The sender normally uses a *location service* to determine the position of the destination node, and to incorporate it in the packet destination address field. Here, the routing process at each node is based on the destination's location available in the packet and the location of the forwarding node's neighbors.
As we can see, position based routing does not require establishment or maintenance of routes, but this usually comes at the expense of an extra hardware. As a further enhancement, position-based routing supports the delivery of packets to all nodes in a given geographical region in a natural way, and this is called *geocasting* which is discussed in the next chapter.
In the following sections we elaborate on the most prominent protocols under each of these categories.

TOPOLOGY-BASED ROUTING PROTOCOLS

PROACTIVE AND REACTIVE ROUTING PROTOCOLS:

Ad hoc routing protocols can be broadly classified as being Proactive (or table-driven) or Reactive (on-demand). Proactive protocols mandates that nodes in a MANET should keep track of routes to all possible destinations so that when a packet needs to be forwarded, the route is already known and can be immediately used. On the other hand, reactive protocols employ a lazy approach whereby nodes only discover routes to destinations on demand, i.e., a node does not need a route to a destination until that destination is to be the sink of data packets sent by the node. Proactive protocols have the advantage that a node experiences minimal delay whenever a route is needed as a route is immediately selected from the routing table. However, proactive protocols may not always be appropriate as they continuously use a substantial fraction of the network capacity to maintain the routing information current. To cope up with this shortcoming, reactive protocols adopt the inverse approach by finding a route to a destination only when needed. Reactive protocols often consume much less bandwidth than proactive protocols, but the delay to determine a route can be significantly high and they will typically experience a long delay for discovering a route to a destination prior to the actual communication. In brief, we can conclude that no protocol is suited for all possible environments, while some proposals using a hybrid approach have been suggested.

UNICAST ROUTING PROTOCOLS

PROACTIVE ROUTING APPROACH

In this section, we consider some of the important proactive routing protocols.

DESTINATION-SEQUENCED DISTANCE-VECTOR PROTOCOL

The destination-sequenced distance-vector (DSDV) [Perkins 1994] is a proactive hop-by-hop distance vector routing protocol, requiring each node to periodically broadcast routing updates. Here, every mobile node in the network maintains a routing table for all possible destinations within the network and the number of hops to each destination. Each entry is marked with a sequence number assigned by the destination node. The sequence numbers enable the mobile nodes to distinguish stale routes from new ones, thereby avoiding the formation of routing loops. Routing table updates are periodically transmitted throughout the network in order to maintain consistency in the table. To alleviate the potentially large amount of network update traffic, route updates can employ two possible types of packets: full dumps or small increment packets. A full dump type of packet carries all available routing information and can require multiple network protocol data units (NPDUs). These packets are transmitted infrequently during periods of occasional movement. Smaller incremental packets are used to relay only the information that has changed since the last full dump. Each of these broadcasts should fit into a standard-size NPDU, thereby decreasing the amount of traffic generated. The mobile nodes maintain an additional table where they store the data sent in the incremental routing information packets. New route broadcasts contain the address of the destination, the number of hops to reach the destination, the sequence number of the information received regarding the destination, as well as a new sequence number unique to the broadcast. The route labeled with the most recent sequence number is always used. In the event that two updates have the same sequence number, the route with the smaller metric is used in order to optimize (shorten) the path. Mobiles also keep track of settling time of the routes, or the weighted average time that routes to a destination could fluctuate before the route with the best metric is received. By delaying the broadcast of a routing update by the length of the settling time, mobiles can reduce network traffic and optimize routes by eliminating those broadcasts that would occur if a better route could be discovered in the very near future.

Note that each node in the network advertises a monotonically increasing sequence number for itself. The consequence of doing it so is that when a node B decides that its route to a destination D is broken, it advertises the route to D with an infinite metric and a sequence number one greater than its sequence number for the route that has broken (making an odd sequence number). This causes any node A routing packets through B to incorporate the infinite-metric route into its routing table until node A hears a route to D with a higher sequence number. **1.1.2**

<u>THE WIRELESS ROUTING PROTOCOL:</u>

The Wireless Routing Protocol (WRP) [Murthy 1996] described is a table-based protocol with the goal of maintaining routing information among all nodes in the network. Each node in the network is responsible for maintaining four tables: Distance table, Routing table, Link-cost table, and the Message Retransmission List (MRL) table. Each entry of the MRL contains the sequence number of the update message, a re-transmission counter, an acknowledgment-required flag vector with one entry per neighbour, and a list of updates sent in the update message. The MRL records which updates in an update message need to be retransmitted and neighbours should acknowledge the retransmission. Mobiles inform each other of link changes through the use of update messages. An update message is sent only between neighbouring nodes and contains a list of updates (the destination, the distance to the destination, and the predecessor of the destination), as well as a list of responses indicating which mobiles should acknowledge (ACK) the update. After processing updates from neighbours or detecting a change in a link, mobiles send update messages to a neighbour. In the event of the loss of a link between two nodes, the nodes send update messages to their neighbours. The neighbours then modify their distance table entries and check for new possible paths through other nodes. Any new paths are relayed back to the original nodes so that they can update their tables accordingly.

Nodes learn about the existence of their neighbours from the receipt of acknowledgments and other messages. If a node is not sending messages, it must send a hello message within a specified time period to ensure connectivity. Otherwise, the lack of messages from the node indicates the failure of that link; this may cause a false alarm. When a mobile receives a hello message from a new node, that new node is added to the mobiles routing table, and the mobile sends the new node a copy of its routing table information.

Part of the novelty of WRP stems from the way in which it achieves freedom from loops. In WRP, routing nodes communicate the distance and second-to-last hop information for each destination in the wireless networks. WRP belongs to the class of path-finding algorithms with an important exception. It avoids the "count-to-infinity" problem by forcing each node to perform consistency checks of predecessor information reported by all its neighbour's. This ultimately (although not instantaneously) eliminates looping situations and provides faster route convergence when a link failure occurs.

REACTIVE ROUTING APPROACH

In this section, we describe some of the most cited reactive routing protocols.

DYNAMIC SOURCE ROUTING

The Dynamic Source Routing (DSR) [Johnson 1996] algorithm is an innovative approach to routing in a MANET in which nodes communicate along paths stored in source routes carried by the data packets. It is referred as one of the purest examples of an on-demand protocol [Perkins 2001]. In DSR, mobile nodes are required to maintain route caches that contain the source routes of which the mobile is aware. Entries in the route cache are continually updated as new routes are learned. The protocol consists of two major phases: route discovery and route maintenance. When a mobile node has a packet to send to some destination, it first consults its route cache to determine whether it already has a route to the destination. If it has an unexpired route to the destination, it will use this route to send the packet. On the other hand, if the node does not have such a route, it initiates route discovery by broadcasting a route request packet. This route request contains the address of the destination, along with the source node's address and a unique identification number. Each node receiving the packet checks whether it knows of a route to the destination. If it does not, it adds its own address to the route record of the packet and then forwards the packet along its outgoing links.

To limit the number of route requests propagated on the outgoing links of a node, a mobile node only forwards the route request if the request has not yet been seen by the mobile and if the mobile's address does not already appear in the route record.

A route reply is generated when the route request reaches either the destination itself, or an intermediate node that contains in its route cache an unexpired route to the destination. By the time the packet reaches either the destination or such an intermediate node, it contains a route record yielding the sequence of hops taken. Figure 2(a) illustrates the formation of the route record as the route request propagates through the network. If the node generating the route reply is the destination, it places the route record contained in the route request into the route reply. If the responding node is an intermediate node, it appends its cached route to the route record and then generates the route reply. To return the route reply, the responding node must have a route to the initiator. If it has a route to the initiator in its route cache, it may use that route. Otherwise, if symmetric links are supported, the node may reverse the route in the route record. If symmetric links are not supported, the node may initiate its own route discovery and piggyback the route reply on the new route request. Figure 2(b) shows the transmission of route record back to the source node.

Route maintenance is accomplished through the use of route error packets and acknowledgments. Route error packets are generated at a node when the data link layer encounters a fatal transmission problem. When a route error packet is received, the hop in error is removed from the node's route cache and all routes containing the hop are truncated at that point. In addition to route error messages, acknowledgments are used to verify the correct operation of the route links. These include passive acknowledgments, where a mobile is able to hear the next hop forwarding the packet along the route.

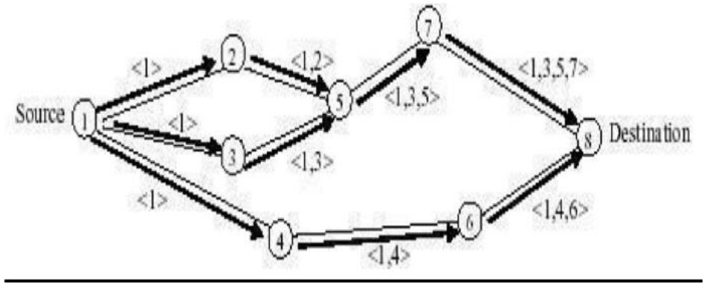


Figure 2(a) – Route discovery in DSR

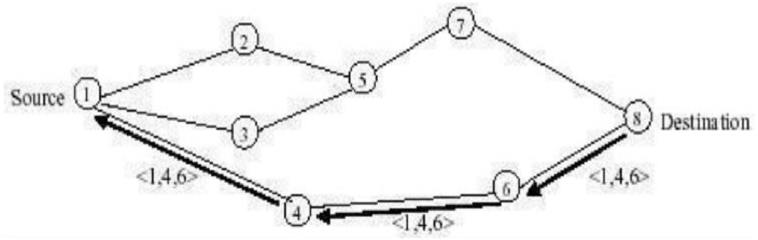


Figure 2(b) – Propagation of route reply in DSR

1.4.2. THE AD HOC ON-DEMAND DISTANCE VECTOR PROTOCOL

The Ad Hoc On-Demand Distance Vector (AODV) routing protocol [Perkins 1999] is basically a combination of DSDV and DSR. It borrows the basic on-demand mechanism of Route Discovery and Route Maintenance from DSR, plus the use of hop-by-hop routing, sequence numbers, and periodic beacons from DSDV. AODV minimizes the number of required broadcasts by creating routes on an on-demand basis, as opposed to maintaining a complete list of routes as in the DSDV algorithm. Authors of AODV classify it as a pure on-demand route acquisition system since nodes that are not on a selected path, do not maintain routing information or participate in routing table exchanges. It supports only symmetric links with two different phases:

• Route Discovery, Route Maintenance; and
• Data forwarding.

When a source node desires to send a message and does not already have a valid route to the destination, it initiates a path discovery process to locate the corresponding node. It broadcasts a route request (RREQ) packet to its neighbours, which then forwards the request to their neighbours, and so on, until either the destination or an intermediate node with a "fresh enough" route to the destination is located. Figure 3(a) illustrates the propagation of the broadcast RREQs across the network. AODV utilizes destination sequence numbers to ensure all routes are loop-free and contain the most recent route information. Each node maintains its own sequence number, as well as a broadcast ID. The broadcast ID is incremented for every RREQ the node initiates, and together with the node's IP address, uniquely identifies an RREQ. Along with the node's sequence number and the broadcast ID, the RREQ includes the most recent sequence number it has for the destination. Intermediate nodes can reply to the RREQ only if they have a route to the destination whose corresponding destination sequence number is greater than or equal to that contained in the RREQ.

During the process of forwarding the RREQ, intermediate nodes record in their route tables the address of the neighbour from which the first copy of the broadcast packet is received, thereby establishing a reverse path. If additional copies of the same RREQ are later received, these packets are discarded. Once the RREQ reaches the destination or an intermediate node with a fresh enough route, the destination/intermediate node responds by unicasting a route reply (RREP) packet back to the neighbour from which it first received the RREQ (Figure 3(b)). As the RREP is routed back along the reverse path, nodes along this path set up forward route entries in their route tables that point to the node from which the RREP came. These forward route entries indicate the active forward route. Associated with each route entry is a route timer which causes the deletion of the entry if it is not used within the specified lifetime. Because the RREP is forwarded along the path established by the RREQ, AODV only supports the use of symmetric links.

Routes are maintained as follows. If a source node moves, it is able to reinitiate the route discovery protocol to find a new route to the destination. If a node along the route moves, its upstream neighbour notices the move and propagates a link failure notification message (an RREP with infinite metric) to each of its active upstream neighbours to inform them of the breakage of that part of the route. These nodes in turn propagate the link failure notification to their upstream neighbours, and so on until the source node is reached. The source node may then choose to re-initiate route discovery for that destination if a route is still desired. An additional aspect of the protocol is the use of hello messages, periodic local broadcasts by a node to inform each mobile node of other nodes in its neighbourhood. Hello messages can be used to maintain the local connectivity of a node. However, the use of hello messages may not be required at all times. Nodes listen for re-transmission of data packets to ensure that the next hop is still within reach. If such a re-transmission is not heard, the node may use one of a number of techniques, including the use of hello messages themselves, to determine whether the next hop is within its communication range. The hello messages may also list other nodes from which a mobile node has recently heard, thereby yielding greater knowledge of network connectivity.

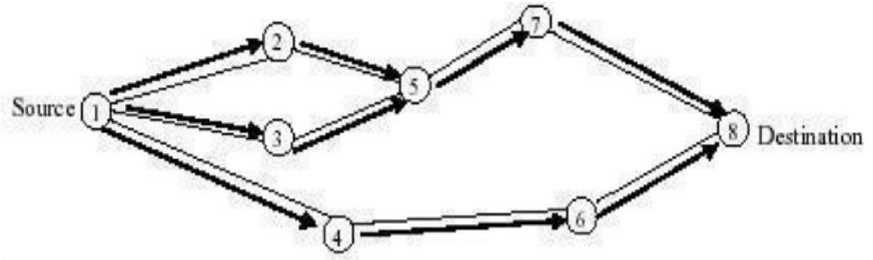


Figure 3(a) – Propagation of RREQ in AODV

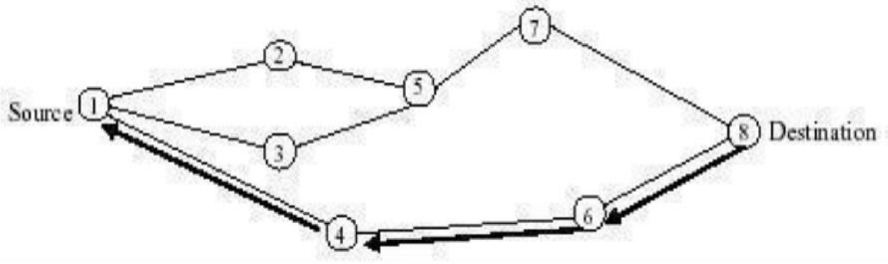


Figure 3(b) – Path taken by the RREP in AODV

1.4.3 Link Reversal Routing and TORA

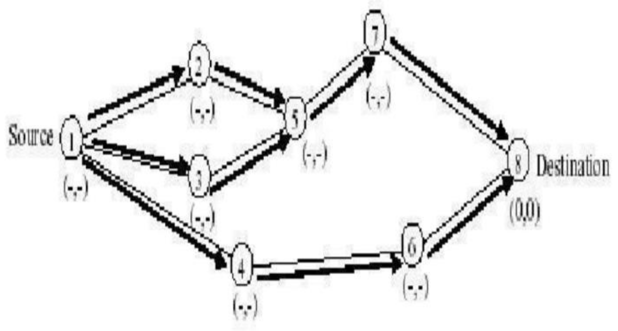
The Temporally Ordered Routing Algorithm (TORA) [Park 1997] is a highly adaptive loop-free distributed routing algorithm based on the concept of link reversal. It is designed to minimize reaction to topological changes. A key design concept in TORA is that it decouples the generation of potentially far-reaching control messages from the rate of topological changes. Such messaging is typically localized to a very small set of nodes near the change without having to resort to a dynamic, hierarchical routing solution with its added complexity. Route optimality (shortest-path) is considered of secondary importance, and longer routes are often used if discovery of newer routes could be avoided. TORA is also characterized by a multipath routing capability. The actions taken by TORA can be described in terms of water flowing downhill towards a destination node through a network of tubes that models the routing state of the real network. The tubes represent links between nodes in the network, the junctions of tubes represent the nodes, and the water in the tubes represents the packets flowing towards the destination. Each node has a height with respect to the destination that is computed by the routing protocol. If a tube between nodes A and B becomes blocked such that water can no longer flow through it, the height of A is set to a height greater than that of any of its remaining neighbours, such that water will now flow back out of A (and towards the other nodes that had been routing packets to the destination via A). Figure 4 illustrates the use of the height metric. It is simply the distance from the destination node. TORA is proposed to operate in a highly dynamic mobile networking environment. It is source initiated and provides multiple routes for any desired source/destination pair. To accomplish this, nodes need to maintain routing information about adjacent (one-hop) nodes. The protocol performs three basic functions:

• Route creation,
• Route maintenance, and
• Route erasure.

For each node in the network, a separate directed acyclic graph (DAG) is maintained for each destination. When a node needs a route to a particular destination, it broadcasts a QUERY packet containing the address of the destination for which it requires a route. This packet propagates through the network until it reaches either the destination, or an intermediate node having a route to the destination. The recipient of the QUERY then broadcasts an UPDATE packet listing its height with respect to the destination. As this packet propagates through the network, each node that receives the UPDATE sets its height to a value greater than the height of the neighbor from which the UPDATE has been received. This has the effect of creating a series of directed links from the original sender of the QUERY to the node that initially generated the UPDATE. When a node discovers that a route to a destination is no longer valid, it adjusts its height so that it is a local maximum with respect to its neighbors and transmits an UPDATE packet. If the node has no neighbors of finite height with respect to this destination, then the node instead attempts to discover a new route as described above. When a node detects a network partition, it generates a CLEAR packet that resets routing state and removes invalid routes from the network. TORA is layered on top of IMEP, the Internet MANET Encapsulation Protocol [Corson 1997], which is required to provide reliable, in-order delivery of all routing control messages from a node to each of its neighbors, plus notification to the routing protocol whenever a link to one of its neighbors is created or broken. To reduce overhead, IMEP attempts to aggregate many TORA and IMEP control messages (which IMEP refers to as objects) together into a single packet (as an object block) before transmission. Each block carries a sequence number and a response list of other nodes from which an ACK has not yet been received, and only those nodes acknowledge the block when receiving it; IMEP retransmits each block with some period, and continues to retransmit it if needed for some maximum total period, after which TORA is notified of each broken link to unacknowledged nodes. For link status sensing and maintaining a list of a node‟s neighbors, each IMEP node periodically transmits a BEACON (or “BEACON-equivalent”) packet, which is answered by each node hearing it with a HELLO (or “HELLO-equivalent”) packet.

As we mentioned earlier, during the route creation and maintenance phases, nodes use the “height” metric to establish a DAG rooted at the destination. Thereafter, links are assigned a direction (upstream or downstream) based on the relative height metric of neighboring nodes as shown in Figure 5(a). In times of node mobility the DAG route is broken, and route maintenance is necessary to reestablish a DAG rooted at the same destination. As shown in Figure 5(b), upon failure of the last downstream link, a node generates a new reference level that effectively coordinates a structured reaction to the failure. Links are reversed to reflect the change in adapting to the new reference level. This has the same effect as reversing the direction of one or more links when a node has no downstream links. Timing is an important factor for TORA because the “height” metric is dependent on the logical time of a link failure; TORA assumes that all nodes have synchronized clocks (accomplished via an external time source such as the Global Positioning System). TORA‟s metric is a quintuple comprising five elements, namely:

• Logical time of a link failure,
• The unique ID of the node that defined the new reference level,
• A reflection indicator bit,
• A propagation ordering parameter & the unique ID of the node.



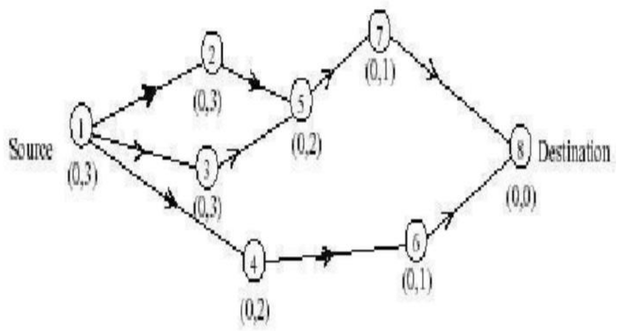


Figure 5(a) – Propagation of the query message
Figure 5(b) – Node's height updated as a result of the update message

The first three elements collectively represent the reference level. A new reference level is defined each time a node loses its last downstream link due to a link failure. TORA‟s route erasure phase essentially involves flooding a broadcast clear packet (CLR) throughout the network to erase invalid routes. In TORA, there is a potential for oscillations to occur, especially when multiple sets of coordinating nodes are concurrently detecting partitions, erasing routes, and building new routes based on each other (Figure 6). Because TORA uses inter-nodal coordination, its instability is similar to the "count-to-infinity" problem, except that such oscillations are temporary and route convergence ultimately occurs. Note that TORA is partially proactive and partially reactive. It is reactive in the sense that route creation is initiated on demand. However, route maintenance is done on a proactive basis such that multiple routing options are available in case of link failures.

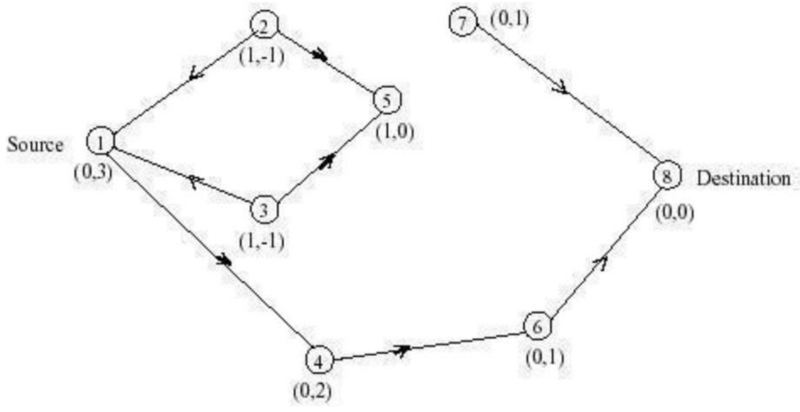


Figure 6 – Route maintenance in TORA

<u>4. ROUTING USING LOCATION INFORMATION:</u>

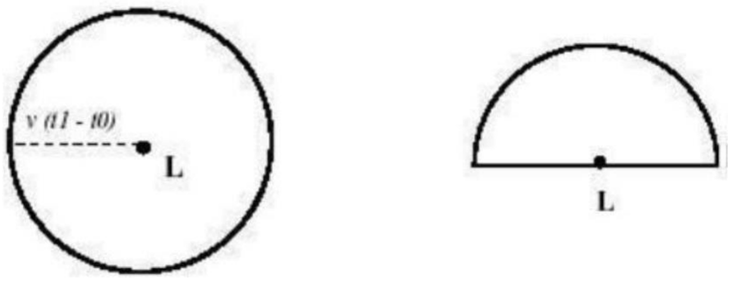
In this section we discuss some ad hoc routing protocols that take advantage of some sort of location information in the routing process.

<u>LOCATION-AIDED ROUTING:</u>

The Location-Aided Routing (LAR) [Ko 1998] protocol exploits location information to limit the scope of route request flood employed in protocols such as AODV and DSR. Such location information can be obtained through GPS (Global Positioning System). LAR limits the search for a route to the so-called request zone, determined based on the expected location of the destination node at the time of route discovery. Two concepts are important to understand how LAR works: Expected Zone and Request Zone.

<u>Expected Zone.</u> Consider a node S that needs to find a route to node D. Assume that node S knows that node D was at location L at time t0, and that the current time is t1. Then, the ―expected zone‖ of node D, from the viewpoint of node S at time t1, is the region expected to contain node D. Node S can determine the expected zone based on the knowledge that node D was at location L at time t0. For instance, if node S knows that node D travels with average speed v, then S may assume that the expected zone is the circular region of radius v(t1

- t0), centered at location L (see Figure 7(a)). If actual speed happens to be larger than the average, then the destination may actually be outside the expected zone at time t1. Thus, expected zone is only an estimate made by node S to determine a region that potentially contains D at time t1. If node S does not know a previous location of node D, then node S cannot reasonably determine the expected zone (the entire region that may potentially be occupied by the ad hoc network is assumed to be the expected zone). In this case, LAR reduces to the basic flooding algorithm. In general, having more information regarding mobility of a destination node can result in a smaller expected zone. For instance, if S knows that destination D is moving north, then the circular expected zone in Figure 7(a) can be reduced to the semi-circle of Figure 7(b).



(a) (b) Figure 7 – Examples of expected zone

Based on the expected zone, we can define the request zone. Again, consider node S that needs to determine a route to node D. The proposed LAR algorithms use flooding with one modification. Node S defines (implicitly or explicitly) a request zone for the route request. A node forwards a route request only if it belongs to the request zone (unlike the flooding algorithm in AODV and DSR). To increase the probability that the route request will reach node D, the request zone should include the expected zone (described above). Additionally, the request zone may also include other regions around the request zone.
Based on this information, the source node S can thus determine the four corners of the expected zone. S includes their coordinates with the route request message transmitted when initiating route discovery. When a node receives a route request, it discards the request if the node is not within the rectangle specified by the four corners included in the route request. For instance, in Figure 8, if node I receives the route request from another node, node I forwards the request to its neighbors, because I determines that it is within the rectangular request zone. However, when node J receives the route request, node J discards the request, as node J is not within the request zone (see Figure 8).

The algorithm just described in the called LAR scheme 1. The LAR scheme 2 is a slight modification to include two pieces of information within the route request packet: assume that node S knows the location (Xd; Yd) of node D at some time t0 – the time at which route discovery is initiated by node S is t1, where t1 ≥ t0. Node S calculates its distance from location (Xd; Yd), denoted as DISTS, and includes this distance with the route request message. The coordinates (Xd; Yd) are also included in the route request packet. With this information, a given node J forwards a route request forwarded by I (originated by node S), if J is within an expected distance from (Xd; Yd) than node I.

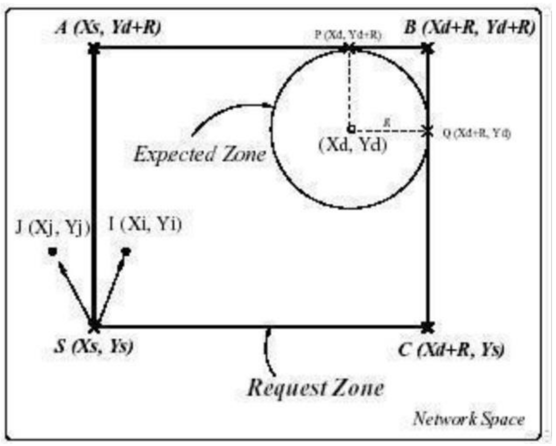


Figure 8 – LAR scheme

.2 DISTANCE ROUTING EFFECT ALGORITHM FOR MOBILITY:

DREAM (Distance Routing Effect Algorithm for Mobility) [Basagni 1998] is a routing protocol for ad hoc networks built around two novel observations. One, called the distance effect, uses the fact that the greater the distance separating two nodes, the slower they appear to be moving with respect to each other. Accordingly, the location information in routing tables can be updated as a function of the distance separating nodes without compromising the routing accuracy. The second idea is that of triggering the sending of location updates by the moving nodes autonomously, based only on a node‟s mobility rate.Intuitively, it is clear that in a directional routing algorithm, routing information about the slower moving nodes needs to be updated less frequently than that about highly mobile nodes. In this way each node can optimize the frequency at which it sends updates to the networks and correspondingly reduce the bandwidth and energy used, leading to a fully distributed and self-optimizing system. Based on these routing tables, the proposed directional algorithm sends messages in the "recorded direction" of the destination node, guaranteeing delivery by following the direction with a given probability.

RELATIVE DISTANCE MICRO-DISCOVERY AD HOC ROUTING

The RDMAR (Relative Distance Micro-discovery Ad Hoc Routing) routing protocol [Aggelou 1999] is a highly adaptive, efficient and scalable routing protocol. It is well-suited in large mobile networks whose rate of topological changes is moderate. A key concept in its design is that protocol reaction to link failures is typically localized to a very small region of the network near the change. This desirable behavior is achieved through the use of a novel mechanism for route discovery, called Relative Distance Micro-discovery (RDM). The concept behind RDM is that a query flood can be localized by knowing the relative distance (RD) between two terminals. To accomplish this, every time a route search between the two terminals is triggered, an iterative algorithm calculates an estimate of their RD, given an average nodal mobility and information about the elapsed time since they last communicated and their previous RD. Based on the newly calculated RD, the query flood is then localized to a limited region of the network centered at the source node of the route discovery and with maximum propagation radius that equals to the estimated relative distance. This ability to localize query flooding into a limited area of the network serves to minimize routing overhead and overall network congestion.

In RDMAR, calls are routed between the stations of the network by using routing tables which are stored at each station of the network; each node is treated as a host as well as a store-and-forward node. Each routing table lists all reachable destinations, wherein for each destination i, additional routing information is also maintained. This includes: the "Default Router" field that indicates the next hop node through which the current node can reach i, the "RD" field which shows an estimate of the relative distance (in hops) between the node and i, the ‖Time_Last_Update‖ (TLU) field that indicates the time since the node last received routing information for i, a “RT_Timeout” field which records the remaining amount of time before the route is considered invalid, and a “Route Flag” field which declares whether the route to i is active. RDMAR comprises of two main algorithms:

• <u>Route Discovery</u> – When an incoming call arrives at node i for destination node j and there is no route available, i initiates a route discovery phase. Here, i has two options; either to flood the network with a route query in which case the route query packets are broadcast into the whole network, or instead, to limit the discovery in a smaller region of the network, if some kind of location prediction model for j can be established. The former case is straightforward. In the latter case, the source of the route discovery, i, refers to its routing table in order to retrieve information on its previous relative distance with j and the time elapsed since i last received routing information for j. Let us designate this time as tmotion. Based on this information and assuming a moderate velocity, Micro_Velocity, and a moderate transmission range, Micro_Range, node i is then able to estimate its new relative distance to destination node j in terms of actual number of hops. To accomplish this, node i calculates the distance offset of DST (DST_Offset) during tmotion, and “adjusts” the result onto their previous relative distance (RDM_Radius).

• <u>Route Maintenance</u> – An intermediate node i, upon reception of a data packet, first processes the routing header and then forwards the packet to the next hop. In addition, node i sends an explicit message to examine whether a bi-directional link can be established with the previous node. RDMAR, therefore, does not assume bi-directional links but in contrast nodes exercise the possibility of having bi-directional links. In this way, nodes that forward a data packet will always have routing information to send the future acknowledgement back to the source. If node i is unable to forward the packet because there is no route available or a forwarding error occurs along the data path as a result of a link or node failure, i may attempt a number of additional re-transmissions of the same data packet, up to a maximum number of retries. However, if the failure persists, node *i* initiates a Route Discovery procedure.

HYBRID ROUTING PROTOCOLS:

<u>Zone Routing Protocol:</u> Zone Routing Protocol (ZRP) [Haas 1998] is a hybrid example of reactive and proactive schemes. It limits the scope of the proactive procedure only to the node"s local neighborhood, while the search throughout the network, although it is global, can be performed efficiently by querying selected nodes in the network, as opposed to querying all the network nodes. In ZRP, a node proactively maintains routes to destinations within a local neighborhood, which is referred to as a routing zone and is defined as a collection of nodes whose minimum distance in hops from the node in question is no greater than a parameter referred to as zone radius. Each node maintains its zone radius and there is an overlap of neighboring zones. The construction of a routing zone requires a node to first know who its neighbors are. A neighbor is defined as a node that can communicate directly with the node in question and is discovered through a MAC level Neighbor discovery protocol (NDP). The ZRP maintains routing zones through a proactive component called the Intrazone routing protocol (IARP) which is implemented as a modified distance vector scheme. On the other hand, the Interzone routing protocol (IERP) is responsible for acquiring routes to destinations that are located beyond the routing zone. The IERP uses a query-response mechanism to discover routes on demand. The IERP is distinguished from the standard flooding algorithm by exploiting the structure of the routing zone, through a process known as bordercasting. The ZRP provides this service through a component called Border resolution protocol (BRP).
The network layer triggers an IERP route query when a data packet is to be sent to a destination that does not lie within its routing zone. The source generates a route query packet, which is uniquely identified by a

combination of the source node"s ID and request number. The query is then broadcast to all the source"s peripheral nodes. Upon receipt of a route query packet, a node adds its ID to the query. The sequence of recorded node Ids specifies an accumulated route from the source to the current routing zone.If the destination does not appear in the node"s routing zone, the node border casts the query to its peripheral nodes. If the destination is a member of the routing zone, a route reply is sent back to the source, along the path specified by reversing the accumulated route. A node will discard any route query packet for a query that it has previously encountered. An important feature of this route discovery process is that a single route query can return multiple route replies. The quality of these returned routes can be determined based on some metric. The best route can be selected based on the relative quality of the route.

Fisheye State Routing (FSR):

The Fisheye State Routing (FSR) protocol [Iwata 1999] introduces the notion of multi-level fisheye scope to reduce routing update overhead in large networks. Nodes exchange link state entries with their neighbors with a frequency which depends on distance to destination. From link state entries, nodes construct the topology map of the entire network and compute optimal routes. FSR tries to improve the scalability of a routing protocol by putting most effort into gathering data on the topology information that is most likely to be needed soon. Assuming that nearby changes to the network topology are those most likely to matter, FSR tries to focus its view of the network so that nearby changes are seen with the highest resolution in time and changes at distant nodes are observed with a lower resolution and less frequently. It is possible to think the FSR as blurring the sharp boundary defined in the network model used by ZRP.

LANDMARK ROUTING (LANMAR) FOR MANET WITH GROUP MOBILITY:

Landmark Ad Hoc Routing (LANMAR) [Pei 2000] combines the features of FSR and Landmark routing. The key novelty is the use of landmarks for each set of nodes which move as a group (viz., a group of soldiers in a battlefield) in order to reduce routing update overhead. Like in FSR, nodes exchange link state only with their neighbors. Routes within Fisheye scope are accurate, while routes to remote groups of nodes are "summarized" by the corresponding landmarks. A packet directed to a remote destination initially aims at the Landmark; as it gets closer to destination it eventually switches to the accurate route provided by Fisheye. In the original wired landmark scheme [Tsuchiya 1988], the predefined hierarchical address of each node reflects its position within the hierarchy and helps find a route to it. Each node knows the routes to all the nodes within it hierarchical partition. Moreover, each node knows the routes to various "landmarks" at different hierarchical levels. Packet forwarding is consistent with the landmark hierarchy and the path is gradually refined from top-level hierarchy to lower levels as a packet approaches the destination. LANMAR borrows from [Tsuchiya 1988] the notion of landmarks to keep track of logical subnets. A subnet consists of members which have a commonality of interests and are likely to move as a "group" (viz., soldiers in the battlefield, or a group of students from the same class). A "landmark" node is elected in each subnet. The routing scheme itself is modified version of FSR. The main difference is that the FSR routing table contains "all" nodes in the network, while the LANMAR routing table includes only the nodes within the scope and the landmark nodes. This feature greatly improves scalability by reducing routing table size and update traffic overhead. When a node needs to relay a packet, if the destination is within its neighbor scope, the address is found in the routing table and the packet is forwarded directly. Otherwise, the logical subnet field of the destination is searched and the packet is routed towards the landmark for that logical subnet. The packet however does not need to pass through the landmark. Rather, once the packet gets within the scope of the destination, it is routed to it directly. The routing update exchange in LANMAR routing is similar to FSR. Each node periodically exchanges topology information with its immediate neighbors. In each update, the node sends entries within its fisheye scope. It also piggy-backs a distance vector with size equal to the number of logical subnets and thus landmark nodes. Through this exchange process, the table entries with larger sequence numbers replace the ones with smaller sequence numbers.

POSITION-BASED ROUTING:

In this section we discuss some ad hoc routing protocols that take advantage of some sort of location information in the routing process [Mauve2001]. Before delving into the forwarding schemes, it is of paramount importance to discuss the principles and issues behind position-based routing, as well as to look into location services.

1.6.1. PRINCIPLES AND ISSUES:

The philosophy of position-based routing is that it is necessary to determine the location of the destination before a packet can be sent. Generally, a location service takes this responsible. Existing location services can be classified according to how many MHs have the service. This can be either *some* specific nodes or *all* the network nodes. Moreover, each location server may maintain the position of *some* specific nodes or *all* the nodes in the network. In the following discussion on location services, we consider all four possible combinations of some-for-some, some-for-all, all-for-some, and all-for all MHs.

In position-based routing, the forwarding decision by a MH is essentially based on the position of a packet's destination and the position of the node's immediate one-hop neighbor. Clearly, the position of the destination is contained in the header of the packet. If a node happens to know an accurate position of the destination, it may choose to update the position of the packet before forwarding it. The position of the neighbors is typically learned through one-hop broadcasts. These beacons are sent periodically by all nodes and contain the position of the sending node.

Three main packet forwarding schemes can be defined for position based routing:
• Greedy forwarding;
• Restricted directional flooding;
• Hierarchical approaches.

For the first two, a node forwards a given packet to one (greedy forwarding) or more (restricted directional flooding) one-hop neighbors that are located closer to the destination than the forwarding node itself. The selection of the neighbor in the greedy case depends on the optimization criteria of the algorithm. It is fairly obvious that both forwarding strategies may fail if there is no one-hop neighbor that is closer to the destination than the forwarding node itself.

Recovery strategies

The third forwarding strategy is to form a hierarchy in order to scale to a large number of MHs. In this chapter we investigate two representatives of hierarchical routing that use greedy forwarding for wide area routing and non-position based approaches for local area routing. Figure 2.8 depicts the two main building blocks, namely, location service and forwarding strategy, that are required for position-based routing. In addition, we illustrate potential classification criteria for the various existing approaches.

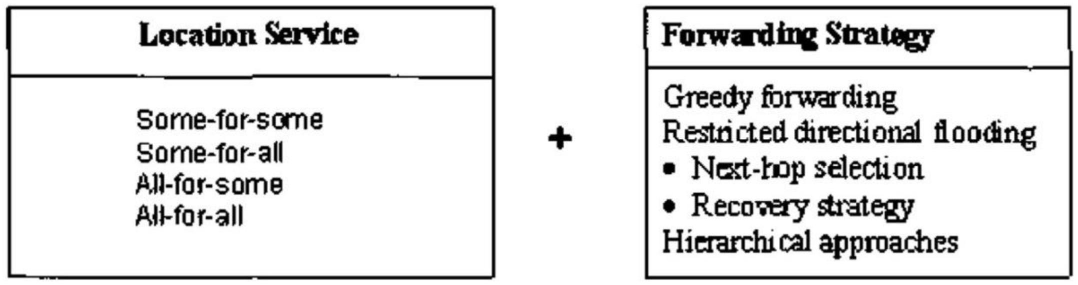


Figure 2.8 - Building blocks for position-based routing [Taken from IEEE Publication Mauve2001]

.2 LOCATION SERVICES:

In order to learn the current position of a specific node, help is needed from a location service. MHs register their current position with this service. When a node does not know the position of a desired communication partner, it contacts the location service and requests that information. In classical one-hop cellular networks, there are dedicated position servers (with well-known addresses) that maintain position information about the nodes in the network. With respect to classification, this is some-for-all approach as the servers are *some* specific nodes, each maintaining the position information about *all* MHs.

In MANETs, such centralized approach is viable only as an eternal service that can be reached via non-ad hoc means. There are two main reasons for this. First, it would be difficult to obtain the location of a position server if the server is a part of the MANET itself. This would represent a chicken-and-egg problem: without the position server it is not possible to get position information, but without the position information the server cannot be reached. Second, since a MANET is dynamic, it might be difficult to guarantee that at least one position server will be present in a given MANET. In the following, we concentrate on decentralized location services that are part of the MANET.

.1 Distance Routing Effect Algorithm for Mobility:

Within Distance Routing Effect Algorithm for Mobility (DREAM) framework [Basagnil998], each node maintains a position database that stores the location information about other nodes that are part of the network. As a consequence, it can be classified as an all-for-all approach. An entry in the position database includes a node identifier, the direction of and distance to the node, as well as a time value that indicates when this information has been generated. Obviously, the accuracy of such an entry depends upon its age. Each node running DREAM periodically floods packets to update the position information maintained by the other nodes. A node can control the accuracy of its position information available to other nodes in two ways:

- By changing the frequency at which it sends position updates. This is known as *temporal resolution;*
- By indicating how far a position update may travel before it is discarded. This is known as *spatial resolution.*

The temporal resolution of sending updates is coupled with the mobility rate of a node, i.e., the higher the speed is, more frequent the updates will be. The spatial resolution is used to provide accurate position information in the direct neighborhood of a node and less accurate information at nodes farther away. The costs associated with accurate position information at remote nodes can be reduced since greater the distance separating two nodes is, slower they appear to be moving with respect to each other. Accordingly, the location information in routing tables can be updated as a function of the distance separating nodes without compromising the routing accuracy. This is called as the *distance effect* and is exemplified by Figure 2.9 where

MH A is assumed stationary, while MHs B and C are moving in the same direction at the same speed. From node A's perspective, the change in direction will be greater for node B than for node C. The distance effect allows low spatial resolution areas far away from the target node, provided that intermediate hops are able to update the position information contained in the packet header. Based on the resulting routing tables, DREAM forwards packets in the *recorded direction* of the destination node, guaranteeing delivery by following the direction with a given probability.

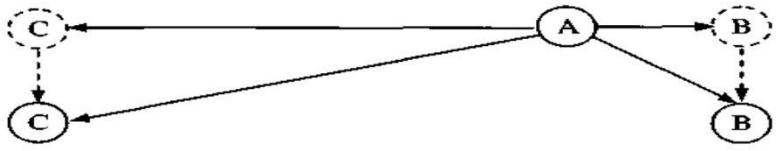


Figure 2.9 – The distance effect in DREAM [Taken from IEEE Publication Mauve2001]

QUORUM-BASED LOCATION SERVICE:

The concept of quorum systems is quite popular in distributed systems and information replication in databases. Here, information updates (write operations) are sent to a subset (quorum) of available nodes, and information requests (read operations) are referred to a potentially different subset. When these subsets are designed such that their intersection is nonempty, it is ensured that an up-to-date version of the sought-after information can always be found. In [Haas 1999], this scheme is employed to develop a location service for MANETs. It is instructive to discuss this scheme through a sample network shown in Figure 2.10. A set of MHs is chosen to host position databases, and this is illustrated by nodes 1-6 in Figure 2.10. Next, a virtual backbone is constructed among the nodes of the subset by utilizing a non-position-based ad hoc routing algorithm.

A MH sends position update messages to the nearest backbone node, which then chooses a quorum of backbone nodes to host the position information. In our example, node D sends its updates to node 6, which might then select quorum A with nodes 1, 2, and 6 to host the information. When a node S wants to obtain the position information, it sends a query to the nearest backbone node, which in turn contacts (through unicast or even multicast) the nodes of a (usually different) quorum. Node 4 might, for example, choose quorum B, consisting of nodes 4, 5, and 6 for the query. Since, by definition, the intersection of two quorum systems is nonempty, the querying node is guaranteed to obtain at least one response with the desired position information. It is important to timestamp position updates. If several responses are received, the one representing the most current position update is selected.

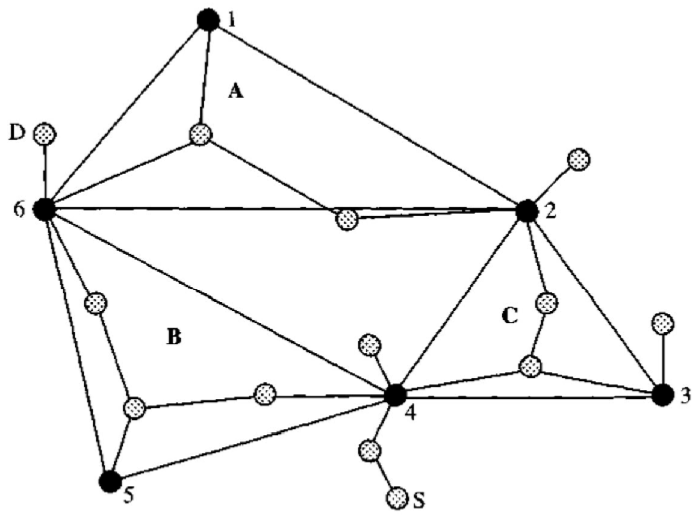


Figure 2.10 – Example of a quorum [Taken from IEEE Publication Mauve2001]

Grid Location Service:

The Grid Location Service (GLS) [Li2000, Morris2000] divides the area that contains the MANET into a hierarchy of squares. In this hierarchy, n-order squares contain exactly *(n* - l)-order squares, forming a so-called *quadtree.* Each node maintains a table of all other nodes
Within the local first-order square. The table is constructed with the help of periodic position broadcasts scoped to the area of the first order square. We present GLS with the assistance of Figure 2.11.

To determine where to store position information, GLS establishes a notion of *near* node IDs, defined as the least ID greater than a node's own ID. When node 10 in Figure 2.11 wants to distribute its position information, it sends position updates to the respective node with the Nearest ID in each of the three surrounding first-order squares. Therefore, the position information is available at nodes 15, 18, 73, and at all nodes that are in the same first-order square as node 10 itself. In the surrounding three second-order squares, again the nodes with the nearest ID are selected to host the node's position (nodes 14, 25, and 29 in the example of Figure 2.11). This process is repeated until the area of the MANET has been fully covered. As we can see, the density of the position information for a given node decreases logarithmically with the distance from that node.

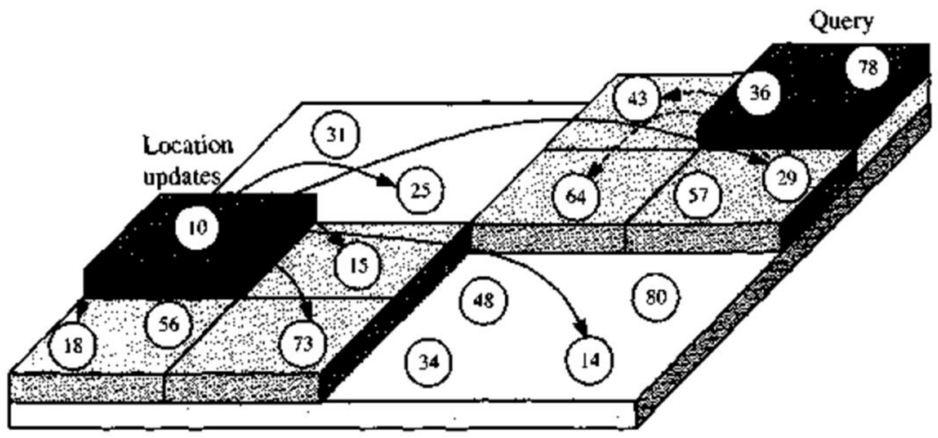


Figure 2.11 – Example of GLS [Taken from IEEE Publication Mauve2001]

OTHER ROUTING PROTOCOLS:

There is plenty of routing protocol proposals for mobile ad hoc networks. Our discussion here is far from being exhaustive. Below we will describe some other routing protocols which employ different optimization criteria as the ones we have previously described.

1.7.1. SIGNAL STABILITY ROUTING:

Another on-demand protocol is the Signal Stability-Based Adaptive Routing protocol (SSR) [Dube 1997]. Unlike the algorithms described so far, SSR selects routes based on the signal strength (weak or strong) between nodes and a node's location stability. The signal strengths of neighboring nodes are obtained by periodic beacons from the link layer of each neighboring node. This route selection criterion of SSR has the effect of choosing routes that have "stronger" connectivity [Chlamtac 1986].

.2 POWER AWARE ROUTING:

In this protocol, power-aware metrics [Singh 1998, Jin 2000] are used for determining routes in wireless ad hoc networks. It has been shown that using these metrics in a shortest-cost routing algorithm reduces the cost/packet of routing packets by 5 - 30 percent over shortest-hop routing (this cost reduction is on top of a 40-70 percent reduction in energy consumption over the MAC layer protocol used). Furthermore, using these new metrics ensures that mean time to node failure is increased significantly, but packet delays do not increase. A recent work [Lee 2000a] concentrates on selecting a route based the traffic and congestion characteristics in the network.

.3 ASSOCIATIVITY-BASED ROUTING:

This is a totally different approach in mobile routing. The Associativity-Based Routing (ABR) [Toh 1997] protocol is free from loops, deadlock, and packet duplicates, and defines a new routing metric for ad hoc mobile networks. In ABR, a route is selected based on a metric that is known as the degree of association stability. Each node periodically generates a beacon to signify its existence. When received by neighboring nodes, this beacon causes their associability tables to be updated. For each beacon received, the associatively tick of the current node with respect to the beaconing node is incremented. Association stability is defined by connection stability of one node with respect to another node over time and space. A high (low) degree of association stability may indicate a low (high) state of node mobility. Associability ticks are reset when the neighbors of a node or the node itself move out of proximity. A fundamental objective of ABR is to derive longer-lived routes for ad hoc networks. The three phases of ABR are:

- Route discovery,
- Route reconstruction (RRC),
- Route deletion.

UNIT-II

DATA TRANSMISSION IN MANETS

Doing network-wide broadcasting in ad hoc networks requires one device to broadcast the information to all its neighbors. For far-away devices, the message is rebroadcasted which could cause collision if multiple device broadcasts the same time and are in the neighborhood. This is also known as the broadcasting storm problem [Nil999] and in this section we discuss ways to perform efficient broadcasting of messages. we assume that MHs in the MANET share a single common channel with carrier sense multiple access (CSMA) [Agrawal2002], but no collision detection (CD) capability (e.g., the IEEE standard 802.11 [IEEE-802.111997]). Synchronization in such a network with mobility is unlikely, and global network topology information is unavailable to facilitate the scheduling of a broadcast. Thus, one straightforward and obvious solution is to achieve broadcasting by flooding (for example, as it is done by mostly all MANET routing algorithms). Unfortunately, as we will see later, it is observed that redundancy, contention, and collision could exist if flooding is done blindly. Several problems arise in these situations including:

- As the radio propagation is omnidirectional and a physical location may be covered by the transmission ranges of several hosts, many rebroadcasts are considered to be redundant
- Heavy contention could exist because rebroadcasting hosts are probably close to each other; and
- As the RTS/CTS handshake (e.g., employed in the IEEE standard (802.11) is inapplicable for broadcast transmissions, collisions are more likely to occur as the timing of rebroadcasts is highly correlated.

BROADCASTING IN A MANET: A MANET consists of a set of MHs that may communicate with one another from time to time, and here no base stations are present. Each host is equipped with a CSMA/CA (carrier sense multiple access with collision avoidance) [Agrawal2002] transceiver. In such an environment, a MH may communicate with each other directly or indirectly. In the latter case, a multi-hop scenario occurs, where the packets originated from the source host are relayed by several intermediate MHs before reaching the destination. The broadcast problem refers to the transmission of a message to all other MHs in the network. The problem we consider has the following characteristics.

• **The broadcast is spontaneous:** Any MH can issue a broadcast operation at any time. For reasons such as the MH mobility and the lack of synchronization, preparing any kind of global topology knowledge is prohibitive (in fact, this is at least as hard as the broadcast problem). Little or no local information may be collected in advance.

• **The broadcast is frequently unreliable:** Acknowledgement mechanism is rarely used. However, attempt should be made to distribute a broadcast message to as many MHs as possible without putting too much effort. The motivations for such an assumption are:

1. A MH may miss a broadcast message because it is off-line, it is temporarily isolated from the network, or it experiences repetitive collisions;
2. Acknowledgements may cause serious medium contention (and thus another "storm") surrounding the Sender

3. In many applications (e.g., route discovery in ad hoc routing Protocols), a 100% reliable broadcast is Unnecessary.

MULTICASTING

In this section, we investigate the problem of multicasting in MANETs where the problem is to broadcast a message to a subset of MANET MHs. We begin by understanding the hard task of multicasting to a group of mobile nodes, together with the various issues behind the design and implementation of a multicast protocol for MANETs. Next, we study the existing multicast protocols for MANETs and show how different they are as compared to broadcasting.

ISSUES IN PROVIDING MULTICAST IN A MANET:

Well-established routing protocols do exist to offer an efficient multicasting service in conventional wired networks. Protocols, designed for fixed networks, may fail to keep up with node movements and frequent topological changes as MHs become increasingly mobile, these protocols need to evolve to cope up with the new environment. But the host mobility increases the protocol overheads substantially.

The broadcast protocols cannot be used either as multicasting requires a selected set of nodes to receive the message while all multicast algorithm depend on the topology of the network and do not consider whether a node belongs to a group or not. Rather, new protocols are being proposed and investigated which take issues such as locations of nodes belonging to a multicast group, and all associated topological changes. Moreover, the nodes of MANET run on batteries, routing protocols must limit the amount of control information that is passed between nodes.

The majority of applications are in the areas where rapid deployment and dynamic reconfiguration are necessary and the wire line network is not available. These include military battlefields, emergency search and rescue sites, classrooms, and conventions where participants share information dynamically using their mobile devices. These applications lend themselves well to multicast operation. In addition, within a wireless medium, it is even more crucial to reduce the transmission overhead and power consumption.

Transient loops may form during reconfiguration of distribution structure (e.g., tree) as a result of mobility. Therefore, reconfiguration scheme should be kept simple to maintain low channel overhead. As we can see, providing an efficient multicasting over MANET faces many challenges including dynamic group membership and constant update of delivery path due to node movement. In the next sections, we cover the major protocols proposed so far and compare them under different criteria.

MULTICAST ROUTING PROTOCOLS:

We can classify the protocols into four categories based on how route to the members of the group is created:

• Tree-Based Approaches

• Meshed-Based Approaches

• Stateless Multicast

• Hybrid Approaches.

1. TREE-BASED APPROACHES:

Most of the schemes for providing multicast in wired network are either source-based or shared tree-based. Different researchers have tried to extend the idea of tree-based approach to provide multicast in a MANET environment. Due to simplicity and innate properties of tree structures, many characteristics can be identified such as: a packet traverses each hop and node in a tree at most once, very simple routing decisions at each node, and the number of copies of a packet is minimized, tree structure built presenting shortest paths amongst nodes, and a loop-free data distribution structure. On the other hand, there are many issues that must be addressed in tree-based approaches. As mentioned earlier, trees provide a unique path between any two nodes. Therefore, having even one link failure could mean reconfiguration of the entire tree structure and could be a major drawback. In addition, multiple packets generated by different sources will require some consideration when utilizing multicast trees such that efficient routing can be established and maintained.

Thus, it is common to consider the use of either a shared tree or establish a separate tree per each source (i.e., separate source trees). As highlighted in [Garcia-Luna-Aceves 1999a], each approach has to deal with own individual issues. For separate source trees, each router (or node in case of MANETs) involved in multiple router groups must maintain a list of pertinent information for each group in which it is involved. Such management per router is inefficient and not scalable. On the other hand, for shared trees, there is a potential that packets may not only not traverse shorter paths, but in fact may be routed on paths with much longer distances than the shortest paths. While any scheme has positive and negative sides, the simple structured coupled with ease of approach has made multicast trees the primary method for realizing multicasting on the Internet. Due to this fact, tree-based approaches for ad hoc networks have been investigated and we will study them in the following sections.

2.MESH-BASED APPROACHES: In contrast to the tree-based approach, mesh-based multicast protocols may have multiple paths between any source and receiver pairs. Existing studies show that tree-based protocols are not necessarily the best suited for multicast in a MANET environment if the network topology changes frequently. In such an environment, mesh-based protocols seem to outperform tree-based proposals due to availability of alternative paths, which allow multicast data grams to be delivered to the receivers even if links fail. The disadvantage of a mesh is the increase in data-forwarding overhead. The redundant forwarding consumes more bandwidth in the bandwidth constrained ad hoc networks. Moreover, the probability of Collision s is higher when a larger number of packets are generated. Therefore, one common problem mesh-based protocols have to consider is how to minimize the data-forwarding overhead caused by flooding. As we shall see, different protocols attack this issue in different ways through the use of forwarding groups, cores, and so on. This section gives an overview of the mesh-based approaches that support multicasting MANETs.

3. STATELESS APPROACHES: Tree-based and mesh-based approaches have an overhead of creating and maintaining the delivery tree/mesh with time. In a MANET environment, frequent movement of MHs considerably increases the overhead in maintaining the delivery tree/mesh. To minimize the effect of such a problem, stateless multicast is proposed wherein a source explicitly mentions the list of destinations in the packet header. Stateless multicast approaches focus on small group multicast and assumes the underlying routing protocol to take care of forwarding the packet to the respective destinations based on the addresses contained in the header. In this section we present the main stateless multicast routing protocols proposed for use in MANETs.

DIFFERENTIAL DESTINATION MULTICAST:

Differential Destination Multicast (DDM) protocol [Ji2001] is meant for small-multicast groups operating in dynamic networks of any size. Unlike other MANET routing protocols, DDM lets source to control multicast group membership. The source encodes multicast receiver addresses in multicast data packets using a special DDM Data Header. This variable length destination list is placed in the packet headers, resulting in packets being self-routed towards the destinations using the underlying unicast routing protocol. It eliminates maintaining per-session multicast forwarding states at intermediate nodes and thus is easily scalable with respect to the number of sessions. DDM supports two kinds of operating modes: "stateless" and "soft state". In stateless mode, the nodes along the data forwarding paths need not maintain multicast forwarding states. An intermediate node receiving a DDM packet only needs to look at the header to decide how to forward the packet. In the "soft-state" mode, based on in-band routing information, each node along the forwarding path remembers the destinations to which the packet has been forwarded last time and its next hop information. By caching this routing information at each node, the protocol does not need to list the entire destination in future data packets. In case changes occur in the underlying unicast routing, an upstream node only needs to inform its downstream nodes about the differences in the destination forwarding since the last packet; hence the name "Differential Destination Multicast". At each node, there is one Forwarding Set (FS) for each multicast session, which records to which destinations this node forwards data. The nodes also maintain a Direction Set (DS) to record the particular next hop to which multicast destination data are forwarded. At the source node, FS contains the same set of nodes as the multicast Member List (ML). In the intermediate nodes, the FS is the union of several subsets based on the data stream received from upstream neighbors. Associated with each set FS_k, there is a sequence number SEQ(FS_k) which is used to record the last DDM Block Sequence Number seen in a received DDM data packet from an upstream neighbor k. It helps to detect loss of data packet containing the forwarding set updates. At a given node, FS also needs to be partitioned into subsets according to the next hops for different destination mechanism defined by the DSR unicast protocol to flood the data packets in the network. Although this is derived from DSR, it can be implemented as a stand-alone protocol. In fact, it does not rely on unicast routing to operate. If DSR has already been implemented on the network, minor modifications are required to enable this protocol. This multicast and broadcast protocol utilizes controlled flooding to distribute data in the network and does not require establishment of a state in the network for data delivery. It is not intended as a general purpose multicast protocol. Its applicability is mainly in environments characterized by very high mobility or by a relatively small number of nodes. In the former case, protocols relying on the establishment of multicast state perform inadequately because they are unable to track the rapid changes in topology. In the latter case, the overhead of keeping multicast state exceeds the overhead of flooding.

DSR SIMPLE MULTICAST AND BROADCAST PROTOCOL:

The DSR Simple Multicast and Broadcast protocol (DSR-MB) [Jetcheva2001b] is designed to provide multicast and broadcast functionality in ad hoc networks. It utilizes the Route Discovery.

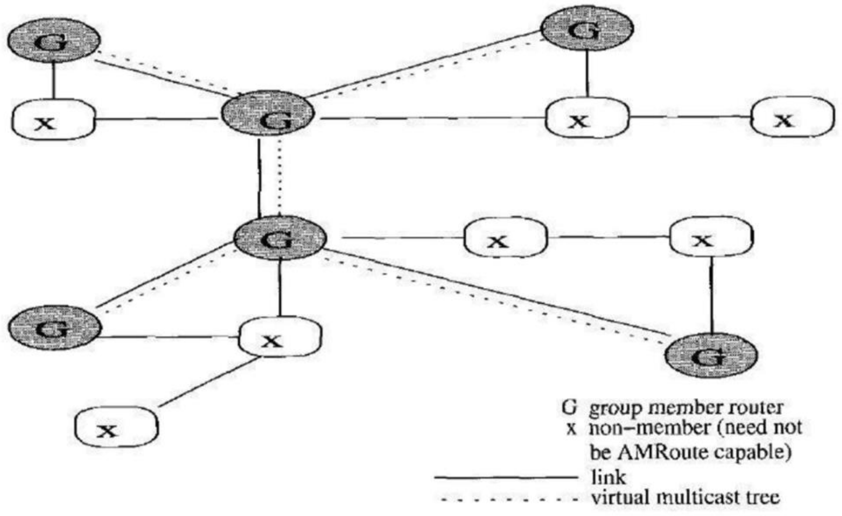
HYBRID APPROACHES: The protocols to provide multicast in ad hoc networks discussed so far, either address efficiency or robustness but not both simultaneously. The tree-based approaches provide high data forwarding efficiency at the expense of low robustness, whereas mesh-based approaches lead to better robustness (link failure may not trigger a reconfiguration) at the expense of higher forwarding overhead and increased network load. Thus, there is a possibility that a hybrid multicasting solution may achieve better performance by combining the advantages of both tree and meshed-based approaches. In this section, we explore the different hybrid approaches to enable ad hoc multicasting.

AD HOC MULTICAST ROUTING PROTOCOL:

The Ad hoc Multicast Routing Protocol (AMRoute) [Bommaiahl998] creates a bi-directional, shared tree by using only group senders and receivers as tree nodes for data distribution. The protocol has two main components: mesh creation and tree setup The mesh creation identifies and designates certain nodes as logical cores and these are responsible for initiating the signaling operation and maintaining the multicast tree to the rest of the group members. A non core node only responds to messages from the core nodes and serves as a passive agent. The selection of logical core in AMRoute is dynamic and can migrate to any other member node, depending on the network dynamics and the group membership. AMRoute does not address network dynamics and assumes the underlying unicast protocol to take care of it. To create a mesh, each member begins by identifying itself as a core and broadcasts JOIN_REQ packets with increasing TTL to discover other members. When a core receives JOIN_REQ from a core in a different mesh for the same group, it replies with a JOIN_ACK. A new bi-directional tunnel is created between the two cores and one of them is selected as core after the mesh merger. Once the mesh has been established, the core initiates the tree creation process. The core sends out periodic TREE_CREATE messages along all links incident on its mesh. Using unicast tunnels, the TREE_CREATE messages are sent only to the group members. Group members receiving non-duplicate TREE_CREATE message forwards it to all mesh links except the incoming one, and marks the incoming and outgoing links as a tree links. If a link is not going to be used as part of the tree, the TREE_CREATE is discarded and TREE_CREATE_NAK is sent back to incoming links. A member node, which wants to leave a group, can do so by sending a JOESLNAK message to its neighboring nodes. AMRoute employs the virtual mesh links to establish the multicast tree, which helps in keeping the multicast delivery tree the same even with the change of network topology as long as routes between core nodes and tree members exist via mesh links. The main disadvantage of this protocol is that it may have temporary loops and may create nonoptimal trees in case of mobility.

MULTICAST CORE-EXTRACTION DISTRIBUTED AD HOC ROUTING:

The Multicast Core-Extraction Distributed Ad hoc Routing (MCEDAR) [Sinhal999] is a multicast extension to the CEDAR architecture. The main idea of MCEDAR is to provide the efficiency of the tree-based forwarding protocols and robustness of mesh-based protocols by combining these two approaches. It is worth pointing out that a source-based forwarding tree is created on a mesh. As such, this ensures that the infrastructure is robust and data forwarding occurs at minimum height trees. MCEDAR decouples the control infrastructure from the actual data forwarding in order to reduce the control overhead.



MOBILITY-BASED HYBRID MULTICAST ROUTING:

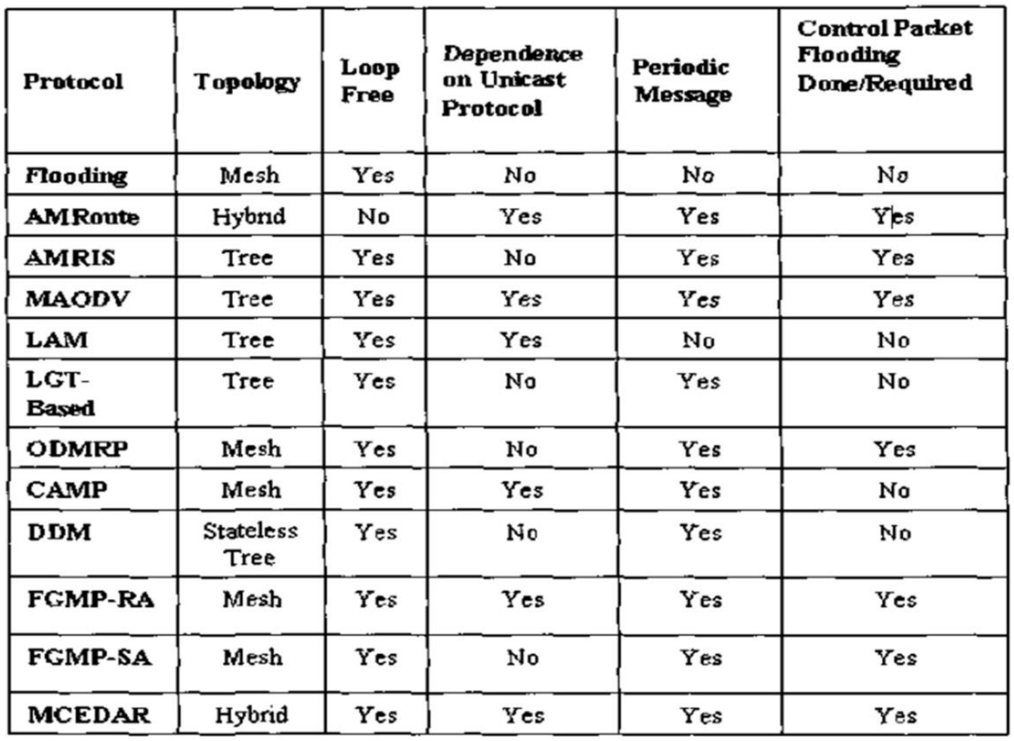
The Mobility-based Hybrid Multicast Routing (MHMR) protocol [An2001] is built on top of the mobility-based clustering infrastructure. In order to deal with the issues of scalability and stability, the structure is hierarchical in nature. The mobility and positioning information is provided via a GPS for each node. For a group of nodes, a cluster-head is chosen to manage and monitor the nodes in a cluster. A mesh structure is built based on all the current clusters.

Thus, MHMR achieves high stability. This is followed by a tree structure built based on the mesh to ensure that the multicasting group achieves maximal efficiency. MHMR also provides a combination of proactive and reactive concepts which enable low route acquisition delay of proactive schemes while achieving low overhead of reactive methods. It is interesting to note that cores are employed in both AM Route and MCEDAR, as well as in many tree and mesh multicast algorithms.

The use of cores has been shown to lower the control overhead. The use of cluster-heads has been proposed in MHMR. This has been shown to be a reasonable approach since dividing the nodes in an ad hoc network into clusters seems to be a promising method in taking care of highly dynamic nodes. Hybrid methods can reveal themselves to be attractive as they can provide protocols that can address further robustness and efficiency. Though hybrid protocols have not been as deeply investigated as tree and mesh protocols, they are under development and recent results indicate its promising future.

GEOCASTING: We now turn our attention to the problem of geocasting over MANETs. As we have mentioned earlier, geocasting is a variant of the conventional multicasting problem and distinguishes itself by specifying hosts as group members within a specified geographical region. In geocasting, the nodes eligible to receive packets are implicitly specified by a physical region and membership changes as mobile nodes move in or out of the geocast region.

Table 3.1 – Comparison of ad hoc multicast routing protocols [Taken from IEEE Publication Cordeiro2003]



Protocol	Topology	Loop Free	Dependence on Unicast Protocol	Periodic Message	Control Packet Flooding Done/Required
Flooding	Mesh	Yes	No	No	No
AMRoute	Hybrid	No	Yes	Yes	Yes
AMRIS	Tree	Yes	No	Yes	Yes
MAODV	Tree	Yes	Yes	Yes	Yes
LAM	Tree	Yes	Yes	No	No
LGT-Based	Tree	Yes	No	Yes	No
ODMRP	Mesh	Yes	No	Yes	Yes
CAMP	Mesh	Yes	Yes	Yes	No
DDM	Stateless Tree	Yes	No	Yes	No
FGMP-RA	Mesh	Yes	Yes	Yes	Yes
FGMP-SA	Mesh	Yes	No	Yes	Yes
MCEDAR	Hybrid	Yes	Yes	Yes	Yes

GEOCAST ROUTING PROTOCOLS:

In this section, we discuss the main geocast routing protocols proposed for use in MANETs. We start with data-transmission oriented protocols, followed by the route creation oriented approaches.

DATA-TRANSMISSION ORIENTED:

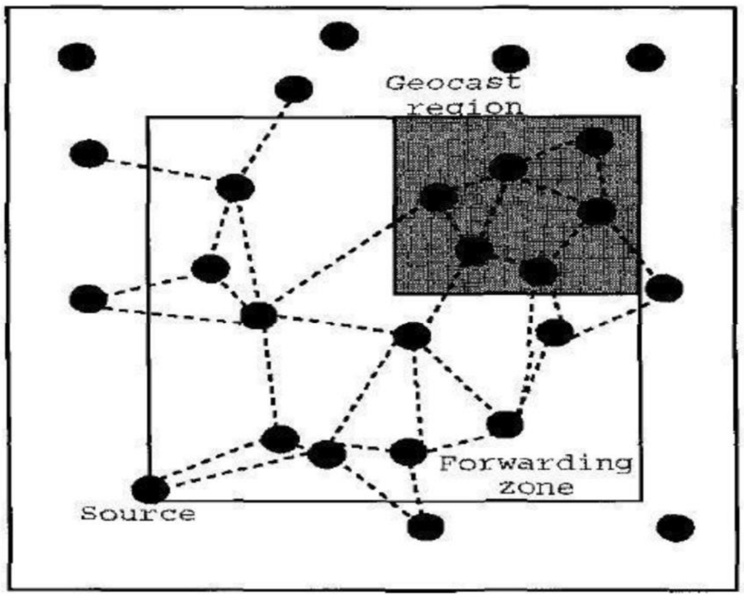
Data-transmission oriented geocast protocols use flooding or a variant of flooding to forward data from the source to the geocast region - and are described here.

LOCATION-BASED MULTICAST:

The Location-Based Multicast (LBM) protocol [Kol999] extends the LAR unicast routing algorithm for geocasting. As we have seen, LAR is an approach to utilize location information to improve the performance (i.e., higher data packet delivery ratio and lower overhead) of a unicast routing protocol in a MANET. Similarly, the goal of LBM is to decrease delivery overhead of geocast packets by reducing the forwarding space for geocast packets, while maintaining accuracy of data delivery. The LBM algorithm is based upon a flooding approach. LBM is essentially identical to flooding data packets, with the modification that a node determines whether to forward a geocast packet further via one of two schemes.

LBM SCHEME 1:

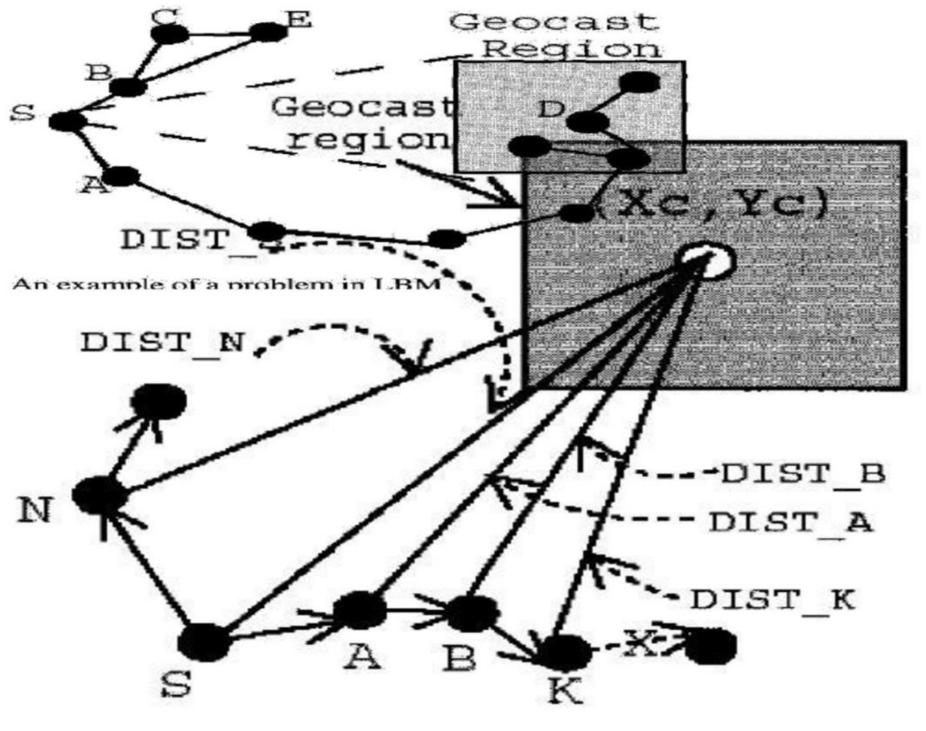
When a node receives a geocast packet, it forwards the packet to its neighbors if it is within *& forwarding zone;* otherwise, it discards the packet. Thus, how to define the forwarding zone becomes the key point of this scheme. Figure 3.11 shows one example of a forwarding zone Boleng2001]. In Figure 3.11, the size of the forwarding zone is dependent on (i) the size of the geocast region and (ii) the location of the sender. In a BOX Forwarding Zone, the smallest rectangle that covers both the source node and the geocast region defines the forwarding zone. All the nodes in the forwarding zone forward data packets to their neighbors. Other kinds of forwarding zones are possible, such as the CONE Forwarding Zone [Boleng2001]. A parameter <5is discussed in [Kol999] to provide additional control on the size of the forwarding zone. When. *S* is positive, the forwarding zone is extended in both positive and negative X and Y directions by <5.(ie., each side increases by 2 *S*).



. BOX forwarding zone

LBM SCHEME 2:

Unlike scheme 1, in which a geocast packet is forwarded based on the forwarding zone, scheme 2 does not have a forwarding zone explicitly. Instead, whether a geocast packet should be forwarded is based on the position of the sender node at the transmission of the packet and the position of the geocast region. That is, for some parameter *8*, node B forwards a geocast packet from node A (originated at node S), if node B is "at least. *S* closer" to the center of the geocast region *(Xc, Yc)* than node A. In other words, *DISTA* > *DISTB*+ *S*. We define *(Xc, Yc)* as the location of the geometrical center of the geocast region, and for any node Z, *DIST z* denotes the distance of node Z from *(Xc, Yc)*. In Figure 3.12 [Kol999], node B will forward a geocast packet transmitted by node A since *DISTA* > *DISTB* and. *d*= 0. Node K will, however, discard a geocast packet transmitted by node B, since node K is not closer to *(Xc, Yc)* than node B. In brief, this protocol ensures that every packet transmission sends the packet closer to the destination. As for the performance, the accuracy (i.e., ratio of the number of geocast group members that actually receive the geocast packets to the number of group members that were supposed to receive the packets) of both LBM schemes is comparable with that of flooding geocast packets throughout the network. However, the number of geocast packets transmitted (a measure of the overhead) is consistently lower for LBM than simple flooding.



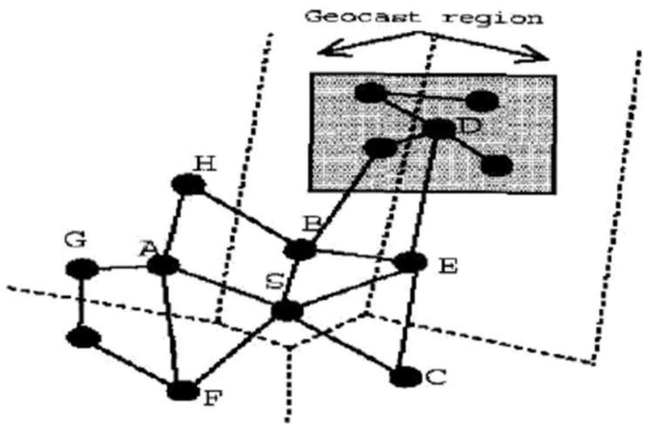
Forwarding zone in LBM scheme 2

VORONOI DIAGRAM BASED GEOCASTING:

The goal of the Voronoi Diagram based Geocasting (VDG) protocol [Stojmenovicl999] is to enhance the success rate and decrease the hop count and flooding rate of LBM. It is observed that the forwarding zone defined in LMB may be a partitioned network between the source node and the geocast region, although there exists a path between the source and the destination. In VDG, the definition of the forwarding zone of LBM has been modified. The neighbors of node A that are located within the forwarding zone in VDG are exactly those neighbors that are closest in the irection of the destination. This definition of a forwarding zone not only resolves the problem of having no nodes inside the forwarding zone, but also precisely determines the expansion of the forwarding zone. This forwarding zone can be implemented with a *Voronoi diagram* for a set of nodes in a given node's neighborhood of a MANET.

A Voronoi diagram of *n* distinct points (i.e., *n* neighbors) in a plane is a partition of the plane into *n* Voronoi regions, which, when associated with node A, consists of all the points in the plane that are the closest to A. In other words, the *Voronoi diagram model* is a model where every point is assigned to a Voronoi region. The subdivision induced by this model is called the *Voronoi diagram* of the set of nodes [Berg]. For example, in Fig, five neighbors of source node S (A, B, C, E and F) carve up the plane into five Voronoi regions. The region associated with node A, consists of nodes G and H, since these two nodes are closer to node A than to any other node. The geocast region is the rectangle with the center D.

In Figure 3.14, the Voronoi regions of nodes B and E intersect the geocast region; thus, only nodes B and E will forward geocast packets from node S. Although there are not any simulations of the VDG algorithm, it is believed that VDG reduces the flooding rates of LBM Scheme 1, as fewer packets should be transmitted. On the other hand, VDG may offer little improvement over LBM Scheme 2, as the end result of the two protocols appears to be similar.



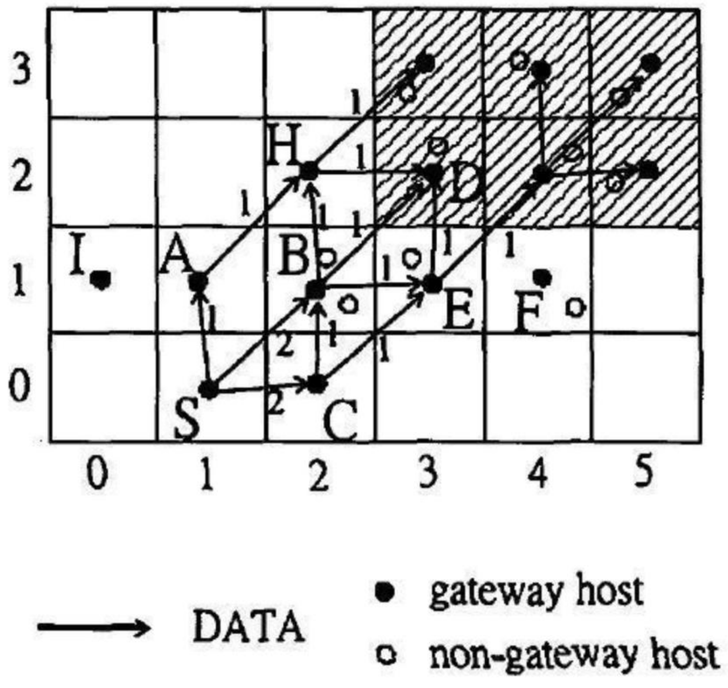
Example of a Voronoi diagram and the request zone

GeoGRID:

Based on the unicast protocol GRID [Liao2001], the GeoGRID protocol [Liao2000] uses location information, which defines the forwarding zone and elects a special host (i.e, *gateway)* in each grid area responsible for forwarding the geocast packets. It is argued in [Liao2000] that the forwarding zone in LBM incurs unnecessary packet transmissions, and a tree-based solution is prohibitive in terms of control overhead. GeoGRID partitions the geographic area of the MANET into two-dimensional (2D) logical grids. Each grid is a square of size *d X d* (there are trade-offs in choosing a good value of *d*, as discussed in [Liao2000].) In GeoGRID, a gateway node is elected within each grid. The forwarding zone is defined by the location of the source and the geocast region. The main difference between GeoGRID, LBM and VDG is the following: in GeoGRID, instead of every node in a forwarding zone transmitting data, only gateway nodes take this responsibility.

There are two schemes on how to send geocast packets in GeoGRID: *Flooding-Based GeoGRID* and *Ticket-Based GeoGRID.* In Flooding-Based GeoGRID, only gateways in every grid within the forwarding zone rebroadcast the received geocast packets. Thus, gateway election becomes the key point of this protocol. In Ticket-Based GeoGRID, the geocast packets are still forwarded by gateway nodes, but not all the gateways in the forwarding zone forward every geocast packet. A total of $m + n$ tickets are created by the source if the geocast region is a rectangle of $m X n$ grids. The source evenly distributes the $m + n$ tickets to the neighboring gateway nodes in the forwarding zone that are closer to the geocast region than the source. A gateway node that receives *X* tickets follows the same procedure as the one defined for the source.

Consider the example where node S begins with five (2+3) tickets. Node S may distribute two tickets to its neighboring nodes A and B, and one ticket to its neighbor node C, which are closer to the geocast region than node S. It is not mentioned in [Liao2000], however, why node C is given fewer tickets than nodes A and B. We believe the philosophy is that each ticket is responsible for carrying one copy of the geocast packet to the geocast region. Hence, if a node is sent a geocast packet that it has seen before, it does not discard it. For example, if node C decides to give its ticket to node B in Figure 3.15, (i.e., node B receives a geocast packet from node C), node B will rebroadcast the packet. In other words, node B will transmit the geocast packet (at least) two times. Both the Flooding-Based GeoGRID and the Ticket-Based GeoGRID protocols need an efficient solution for the gateway election. Once this node is elected, it remains the gateway until it moves out of the grid. One problem of this selection process is when another potential gateway roams closer to the physical center of the grid than the currently assigned gateway and cannot be elected as the gateway until the current gateway leaves the grid. To eliminate this possibility, multiple gateways could be



A geocast example for the Ticket-Based GeoGRID protocol

allowed to reside in a grid temporally. In this situation, if a gateway hears a packet from another gateway at a location closer to the physical center of its grid, it silently turns itself into a non-gateway node and does not forward any further geocast packets. However, if the grid size is small, or the mobility of the node is low, this problem may not be severe. Another effective way of gateway election is via the concept of Node Weight [Basagnil999]. For example, we could assign the weight of a node as being inversely proportional to its speed. Flooding-Based GeoGRID and Ticket-Based GeoGRID have obvious advantages over LBM Scheme 1 and LBM Scheme 2, especially in dense networks. The two GeoGRID protocols should offer both higher accuracy and lower delivery cost than LBM and VDG due to the reduced number of transmitted packets.

GeoTORA:

The goal of the GeoTORA protocol [Ko2000] is to reduce the overhead of transmitting geocast packets via flooding techniques, while maintaining high accuracy. The unicast routing protocol TORA is used by GeoTORA to transmit geocast packets to a geocast region. As TORA is a distributed routing protocol based on a "link reversal" algorithm, it provides multiple routes to a destination. Despite dynamic link failures, TORA attempts to maintain a destination-oriented directed acyclic graph such that each node can reach the destination. In GeoTORA, a source node essentially performs an *anycast* to any geocast group member (i.e, any node in the geocast region) via TORA. When a node in the geocast region receives the geocast packet, it floods the packet such that the flooding is limited to the geocast region. The accuracy of GeoTORA is high, but not as high as pure flooding or LBM. One reason is only one node in the geocast region receives the geocast packet and if that node is partitioned from other nodes in the geocast region, the accuracy reduces.

MESH-BASED GEOCAST ROUTING PROTOCOL:

The Mesh-based Geocast Routing (MGR) protocol [Boleng2001] uses a mesh for geocasting in an ad hoc environment in order to provide redundant paths between the source and the group members. Since the group members in a geocast region are in close proximity to each other, it is less costly to provide redundant paths from a source to a geocast region than to provide the redundant paths from a source to a multicast group of nodes that may not be in close proximity of each other. Instead of flooding geocast packets, the MGR Protocol tries to create redundant routes via control packets. First, the protocol floods JOIN-DEMAND packets in a forwarding zone. A JOIN-DEMAND packet is forwarded in the network until it reaches a node in the geocast region. This node unicasts a JOIN-TABLE packet back to the source by following the reverse route of the JOIN-DEMAND packet. Thus, the nodes on the edge of the geocast region become a part of the mesh. Once the first JOINTABLE packet is received by the source, data packets can be sent to the nodes in the geocast region. Figure shows an example of geocast communication via a mesh.

TCP OVER AD HOC NETWORKS:

Over the past few years, ad hoc networks have emerged as a promising approach for future mobile IP applications. This scheme can operate independently from existing underlying infrastructure and allows simple and fast implementation. Obviously, the more this technology
Evolves, higher will be the probability of being an integral part of the global Internet [Perkins2002]. Therefore, considerable research efforts have been put on the investigation of the Transmission Control Protocol (TCP) [Comer 1995] performance over ad hoc networks.

As we know, TCP is the prevalent transport layer in the IP world today and is employed by a vast majority of applications, especially those requiring reliability. More specifically, TCP is the most widely used transport protocol for data services like file transfer, email, web browsing, and so on. Therefore, it is essential to look at the TCP performance over ad hoc networks. TCP is now fine-tuned to work well in wired environments; however its performance is highly degraded due to the high error rates and longer delays of wireless mediums, as well as mobility.

In recent years, several improvements for TCP have been proposed for cellular [Balakrishnanl997, Zhang2001] wireless networks, but still much work has to be done for the ad hoc networks paradigm.

Unlike cellular networks, where only the last hop is based on a wireless medium, ad hoc networks are composed exclusively of wireless links, where multi-hop connections are often in place. Besides, in an adhoc scenario, all nodes can move freely and unpredictably, which makes the clock based TCP congestion control quite hard. In particular, as the errors in wireless networks occur not only due to congestion but also due to medium constraints and mobility, TCP needs to distinguish the nature of the error so that it can take the most appropriate action. Factors such as path asymmetry (that may also be caused by lower layers strategies, among other elements) and congestion window size also impact the performance of this protocol. These issues and others are addressed in detail in this chapter.

Although there are a number of differences between cellular and ad hoc networks, some of the ideas developed for the former can be used in the latter as well. As a matter of fact, many of the proposed solutions for TCP over ad hoc networks represent a mix of old concepts developed for cellular network properly adapted for this multi-hop scenario. Nevertheless, we cover some solutions specifically tailored for ad hoc networks, while many issues are still open.

TCP PROTOCOL OVERVIEW:

The TCP protocol is defined in the Request for Comment (RFC) standards document number 793 [Postell981] by the Internet Engineering Task Force (IETF) [IETFwww]. The original specification written in 1981 was based on earlier research and experimentation in the
Original ARPANET. It is important to remember that most applications on the Internet make use of TCP, relying upon its mechanisms that ensure safe delivery of data across an unreliable IP layer below. In this section we explore the fundamental concepts behind TCP and how it is
used to transport data between two endpoints. More specifically, we focus on those aspects of TCP that are of importance to understand its performance over ad hoc networks. TCP adds a great deal of functionality to the IP service it is layered over:

- **STREAMS:** TCP data is organized as a stream of bytes, much like a file. The datagram nature of the network is concealed. A mechanism (the Urgent Pointer) exists to let out-of-band data be specially flagged;
- **RELIABLE DELIVERY:** Sequence numbers are used to coordinate whichdata has been transmitted and received. TCP will arrange for retransmission if it determines that data has been lost;
- **NETWORK ADAPTATION:** TCP will dynamically learn the delay characteristics of a network and adjust its operation to maximize throughput without overloading the network;
- **FLOW CONTROL:** TCP manages data buffers, and coordinates traffic so that its buffers will never overflow. Fast senders will be stopped periodically to keep up with slower receivers

DESIGNED AND FINE-TUNED TO WIRED NETWORKS:

The design of TCP has been heavily influenced by what is commonly known as the "end-to-end argument" [Clarkl988]. As it applies to the wired Internet, the system gets unnecessarily complicated by putting excessive intelligence in physical and link layers to handle error control, encryption or flow control. While these functions need to be done at the endpoints anyway, the result is the provision of minimal functionality on a hop-by-hop basis and maximal control between end to- end communicating systems.

TCP performance is often dependent on the flow control and the congestion control. Flow control determines the rate at which data is transmitted between a sender and a receiver. Congestion control defines the methods for implicitly interpreting signals from the network in order for a sender to adjust its rate of transmission. Ultimately, intermediate devices, such as IP routers, would only be able to control congestion. A recent study on congestion control examines the current state of activity [KristofKOOO].

Timeouts and retransmissions handle error control in TCP. The nature of TCP and the underlying packet switched network provide formidable challenges for managers, designers and researchers. In the design of TCP for wireless networks, it is important to incorporate link layer acknowledgements and error detection/correction functionality. Furthermore, when we consider MANETs, the mobility comes into picture. Therefore, high error rates, longer delays, and mobility makes MANET environments extremely challenging to the operation of TCP as it tears down most the assumptions over which TCP was designed.

TCP BASICS:

TCP is often described as a byte stream, connection-oriented, fullduplex, reliable delivery transport layer protocol. In this subsection, we discuss the meaning for each of these descriptive terms.

BYTE STREAM DELIVERY:

TCP interfaces between the application layer above and the network layer below. When an application sends data to TCP, it does so in 8-bit byte streams. It is then up to the sending TCP to segment or delineate the byte stream in order to transmit data in manageable pieces to the receiver. It is this lack of "record boundaries" which gives it the name "byte stream delivery service".

CONNECTION-ORIENTED:

Before two communicating TCP entities (the sender and the receiver) can exchange data, they must first agree upon the willingness to communicate. Analogous to a telephone call, a connection must first be made before two parties exchange information.

<u>FULL-DUPLEX:</u>

No matter what a particular application may be, TCP almost always operates in full-duplex mode. It is sometimes useful to think of a TCP session as two independent byte streams, traveling in opposite directions. No TCP mechanism exists to associate data in the forward and reverse byte streams, and only during connection start and close sequences can TCP exhibit asymmetric behavior (i.e., data transfer in the forward direction but not in the reverse, or vice versa).

<u>RELIABILITY:</u>

A number of mechanisms help provide the reliability TCP guarantees. Each of these is described briefly below.

- **Checksums:** All TCP segments carry a checksum, which is used by the receiver to detect errors with either the TCP header or data;
- **Duplicate data detection:** It is possible for packets to be duplicatedin packet switched network; therefore TCP keeps track of bytes received in order to discard duplicate copies of data that has already been received;
- **Retransmissions:** In order to guarantee delivery of data, TCP must implement retransmission schemes for data that may be lost or damaged. The use of positive acknowledgements by the receiver to the sender confirms successful reception of data. The lack of positive acknowledgements, coupled with a timeout period (see timers below) calls for a retransmission;
- **Sequencing:** In packet switched networks, it is possible for packets to be delivered out of order. It is TCP's job to properly sequence segments it receives so that it can deliver the byte stream data to an application in order;
- **Timers:** TCP maintains various static and dynamic timers on data sent. The sending TCP waits for the receiver to reply with an acknowledgement within a bounded length of time. If the timer expires before receiving an acknowledgement, the sender can retransmit the segment.

<u>TCP HEADER FORMAT:</u>

As we know, the combination of TCP header and TCP in one packet is called a TCP segment. Figure 7.1 depicts the format of all valid TCP segments. The size of the header without options is 20 bytes. Below we briefly define each field of the TCP header.

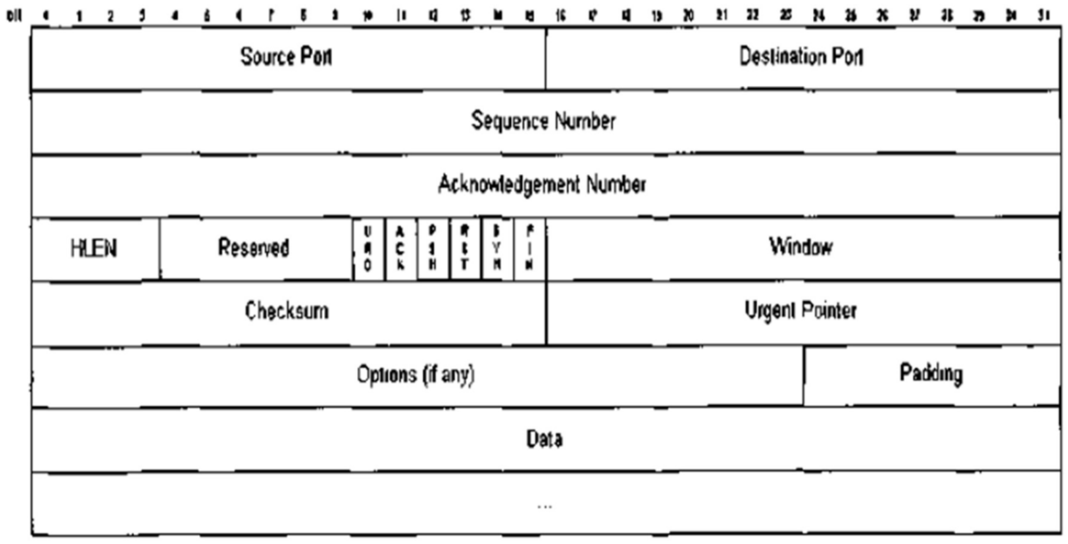


Figure 7.1 – TCP header format

SOURCE PORT: This is a 16-bit number identifying the application where the TCP segment originated from within the sending host. The port numbers are divided into three ranges: well-known ports (0 through 1023), registered ports (1024 through 49151) and private ports (49152 through 65535). Port assignments are used by TCP as an interface to the application layer. For example, the TELNET server is always assigned to
the well-known port 23 by default on TCP hosts. A pair of IP addresses (source and destination) plus a complete pair of TCP ports (source and destination) defines a single TCP connection that is globally unique.
DESTINATION PORT: A 16-bit number identifying the application the TCP segment is destined for on a receiving host. Destination ports use the same port number assignments as those set aside for source ports.
SEQUENCE NUMBER: Within the entire byte stream of the TCP connection, a 32-bit number, identifying the current position of the first data byte in the segment. After reaching 232 -1, this number will wrap around to 0.
ACKNOWLEDGEMENT NUMBER: This is a 32-bit number identifying the next data byte the sender expects from the receiver. Therefore, this number will be one greater than the most recently received data byte. This field is used only when the ACK control bit is turned on.
HEADER LENGTH: A 4-bit field that specifies the total TCP header length in 32-bit words (or in multiples of 4 bytes). Without options, a TCP header is always 20 bytes in length. On the other hand, the largest a TCP header is 60 bytes. Clearly, this field is required because the size of the options field(s) cannot be determined in advance. Note that this field is called "data offset" in the official TCP standard, but header length is more commonly used.
RESERVED: A 6-bit field currently unused and reserved for future use.
CONTROL BITS:
• *Urgent Pointer (URG)* - If this bit field is set, the receiving TCP should interpret the urgent pointer field (see below);
• *Acknowledgement (ACK)* - If this bit is set, the acknowledgment field is valid;
• *Push Function (PSH)* - If this bit is set, the receiver should deliver this segment to the receiving application as soon as possible. An example of its use may be to send a Control-C request to an application, which can jump ahead of queued data;

• *Reset Connection (RST)* - If this bit is present, it signals the receiver that the sender is aborting the connection and all the associated queued data and allocated buffers can be freely relinquished;

• *Synchronize (SYN)* - When present, this bit field signifies that the sender is attempting to "synchronize" sequence
numbers. This bit is used during the initial stages of connection establishment between a sender and a receiver;

• *No More Data from Sender (FIN)* - If set, this bit field tells the receiver that the sender has reached the end of its byte stream for the current TCP connection.

Window: This is a 16-bit integer used by TCP for flow control in the form of a data transmission window size. This number tells the sender how much data the receiver is willing to accept. The maximum value for this field would limit the window size to 65,535 bytes. However, a "window scale" option can be used to make use of even larger windows.

Checksum: A TCP sender computes the checksum value based on the contents of the TCP header and data fields. This 16-bit value will be compared with the value the receiver generates using the same computation. If the values match, the receiver can be very confident that the segment arrived intact.

Urgent Pointer: In certain circumstances, it may be necessary for a TCP sender to notify the receiver of urgent data that should be processed by the receiving application as soon as possible. This 16-bit field tells the receiver when the last byte of urgent data in the segment ends.

Options: In order to provide additional functionality, several optional parameters may be used between a TCP sender and a receiver. Depending on the option(s) used, the length of this field varies in size, but it cannot be larger than 40 bytes due to the maximum size of the header length field (4 bits). The most common option is the maximum segment size (MSS) option where a TCP receiver tells the TCP sender the maximum segment size it is willing to accept. Other options are often used for various flow control and congestion control techniques.

Padding: Because options may vary in size, it may be necessary to "pad" the TCP header with zeroes so that the segment ends on a 32-bit word boundary as defined by the standard.

Data: Although not used in some circumstances (e.g., acknowledgement segments with no data in the reverse direction), this variable length field carries the application data from TCP sender to receiver. This field coupled with the TCP header fields constitutes a TCP segment.

<u>CONGESTION CONTROL</u>

TCP congestion control and Internet traffic management issues in general is an active area of research and experimentation. Although this section is a very brief summary of the standard congestion control algorithms widely used in TCP implementations today, it covers the main points to understand behavior of the TCP over MANETs. These algorithms are defined in [Jacobsonl988] and [Jacobson 1990a], and the most update version of the TCP congestion control algorithm can be found in [Allman 1999].

<u>Slow Start</u>

Slow Start, a requirement for TCP software implementation, is a mechanism used by the sender to control the transmission rate, otherwise known as sender-based flow control. The rate of acknowledgements returned by the receiver determines the rate at which the sender can transmit data. Whenever a TCP connection starts, the Slow Start
algorithm at the sender initializes a *congestion window* (CWND) to one segment. As the connection is carried out and acknowledgements are returned by the receiver, the congestion window increases by one segment for each acknowledgement returned. Thus, the sender can transmit the minimum of the congestion window and the advertised window (contained in the header of the acknowledgment packet) of the receiver, which is simply called the transmission window and is increased exponentially.

<u>Congestion Avoidance</u>

During the initial data transfer phase of a TCP connection, the Slow Start algorithm is used. However, there may be a point during Slow Start that the network is forced to drop one or more packets due to overload or congestion. If this happens, Congestion Avoidance is used to reduce the transmission rate. However, Slow Start is used in conjunction with Congestion Avoidance in order to get the data transfer going again so it does not slow down and stay slow. In the Congestion Avoidance algorithm, the expiration of a timer called *retransmission timeout* (RTO) or the reception of duplicate ACKs implicitly signal the sender that a network congestion situation is occurring. The sender immediately sets its transmission window to one half of the current window size (the minimum of the congestion window and the receiver's advertised window size). If congestion is indicated by a timeout, the congestion window is reset to one segment, which automatically puts the sender into Slow Start mode.

As data is received during Congestion Avoidance, the congestion window is increased. However, Slow Start is only used up to the halfway point where congestion originally occurred. This halfway point was recorded earlier as the new transmission window. After this halfway point, the congestion window is increased by one segment for all segments in the transmission window that are acknowledged. This mechanism will force the sender to slowly grow its transmission rate, as it will approach the point where congestion had previously been detected.

Fast Retransmit

When a duplicate ACK is received, the sender does not know if thisis because a TCP segment was lost or simply because a segment was delayed and received out of order at the receiver. If the receiver can reorder segments, it should not be long before the receiver sends the latest expected acknowledgement. Typically, no more than one or two
duplicate ACKs should be received when a simple out of order conditions takes place. If, however, more than two duplicate ACKs are received by the sender, it is a strong indication that at least one segment has been lost. Here, the TCP sender will assume enough time has lapsed for all segments to be properly re-ordered by the fact that the receiver
had enough time to send three duplicate ACKs.

Thus, whenever three or more duplicate ACKs are received, the sender does not even wait for the RTO to expire and retransmits the segment (as indicated by the position of the duplicate ACK in the byte
stream). This process is called the Fast Retransmit algorithm and was first defined in [Jacobson 1990a]. Immediately following Fast Retransmit,
there is the Fast Recovery algorithm described next.

Fast Recovery

Since the Fast Retransmit algorithm is used when duplicate ACKs are being received, the TCP sender has implicit knowledge that there is data still flowing to the receiver. Why? The reason is because duplicate ACKs can only be generated when a segment is received. This is a strong indication that serious network congestion may not exist and that the lost segment was a rare event. Therefore, instead of reducing the flow of data abruptly by going all the way into Slow Start, the sender only enters

Congestion Avoidance mode. Rather than start at a window of one segment as in Slow Start mode, the sender resumes transmission with a larger window, incrementing as if in Congestion Avoidance mode. This allows for higher throughput under the condition of only moderate congestion [Lin 1998].

Round-Trip Time Estimation

When a host transmits a TCP packet to its peer, it must wait a period of time for an acknowledgment. If the reply does not come within the expected period, the packet is assumed to have been lost and the data is retransmitted. The obvious question - How long do we wait? - lacks a simple answer. Over an Ethernet, no more thana few microseconds should be needed for a reply. If the traffic must flow over the wide-area Internet, a second or two might be reasonable during peak utilization times. If we are talking to an instrument package on a satellite hurtling toward Mars, minutes might be required before a reply. There is no one answer to the question - How long? As a result, all modern TCP implementations seek to answer this question by monitoring the normal exchange of data packets and developing an estimate of how long is "too long". This process is called Round-Trip Time (RTT) estimation which computes one of the most important performance parameters in a TCP exchange, especially when you consider that on an indefinitely large transfer all TCP implementations eventually drop packets and retransmit them, no matter how good the quality of the link is. If the RTT estimate is too low, packets are retransmitted unnecessarily; if too high, the connection can sit idle while the host waits to timeout. If we bring this discussion to ad hoc networks, things become quite complicated. Once a TCP connection is established over multiple wireless hops in an ad hoc network and packets start flowing, TCP initiates the process of RTT estimation. As some nodes may be mobile, some station on the route from the source to the destination may move causing a link breakage. Very often when this happens, the RTO of TCP (set based on the RTT estimation) goes off and triggers TCP's congestion control algorithms. Clearly, this is not what TCP is supposed to do as there is no congestion in place. Rather, the expiration of the RTO was caused by mobility and not by congestion. As we discuss later, most of the current research on TCP over ad hoc concentrates on addressing this issue.

TCP AND MANETS

MANETs pose some tough challenges to TCP primarily because TCP has not been designed to operate in such a highly dynamic environment [Oliveira2002]. In general, we can identify three main different types of challenges posed to TCP over ad hoc networks.

• First, as the topology changes, the path is interrupted and TCP goes into repeated, exponentially increasing time-outs with severe performance impact. Efficient retransmission strategies have been proposed to overcome such problems (e.g., [Chandran2001, Holland 1999a])

• The second problem has to do with the fact that the TCP performance in ad hoc multi-hop environment depends critically on the congestion window in use. If the window grows too large, there are too many packets (and ACKs) on the path, all competing for the same wireless shared medium. Congestion builds up and causes "wastage" of the broadcast medium with consequent throughput degradation [Fu2005, Oliveira2005];

• The third problem is significant TCP unfairness which has been recently revealed and reported through both simulations and testbed measurements [Cordeiro2002].

In reality, even though TCP has evolved significantly over the years toward a robust and reliable service protocol, the focus has been primarily on wired networks. In this scenario, the additive increase multiplicative-decrease strategy coupled with the fast recovery and fast retransmits mechanisms [Allmanl999], inherent in most of current TCP versions; provide an effective congestion control solution.

The key idea of TCP is to probe the network in order to determine the availability of resources. It injects packets at an increasing rate into the network until a packet loss is detected, whereby it infers the network is facing congestion. Then, the TCP sender shrinks its CWND, retransmits the lost packet and resumes transmission at a lower increasing rate. If the losses persist (no timely ACK received), at every retransmission the sender doubles (up to 64s) its wait timer (i.e., the RTO) so that it can wait longer for the ACK of the current packet being transmitted. This is the *exponential backoff* strategy depicted in Figure 7.2. More details on
this mechanism can be found in [Stevens 1994].

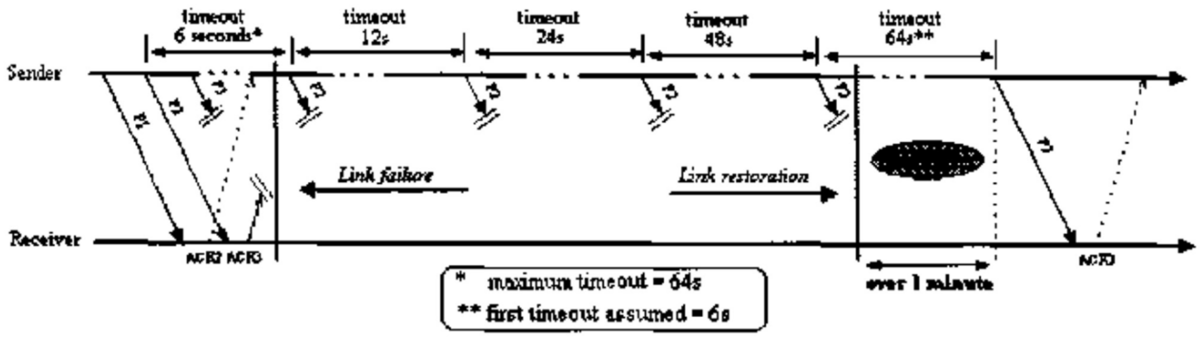


Figure 7.2 – Drawback of the TCP exponential backoff algorithm in MANETs

SOLUTIONS FOR TCP OVER AD HOC

We now present the most prominent schemes that have been specifically proposed to overcome TCP performance problems in ad hoc networks. Here, we classify the proposed solutions into *Mobility-related* and *Fairness-related* based on the key TCP issue they aim to overcome. The mobility-related approaches address the TCP problems resulting from node mobility which may mistakenly trigger TCP congestion control mechanisms. On the other hand, fairness-related solutions tackle the serious unfairness conditions raised when TCP is run over MANETs.

MOBILITY-RELATED

Notably, most of the solutions in this category have limitations which can compromise their widespread deployment. Nevertheless, it is of paramount importance to understand them as they may serve as the basis for future research. In the following we discuss the details of each one of them.

TCP-FEEDBACK

As the name suggests, TCP-Feedback (TCP-F) [CRVP97] is a feedback-based scheme in which the TCP sender can effectively distinguish between route failure and network congestion by receiving Route Failure Notification (RFN) messages from intermediate nodes. The idea is to push the TCP into a "snooze state" whenever such messages are received. In this state, TCP stops sending packets and freezes all its variables such as timers and CWND size, which makes sense once there is no available route to the destination. Upon receipt of a Route Re-establishment Notification (RRN) message from the routing protocol, indicating that there is again an available path to the destination, the sender leaves the frozen state and resumes transmission using the same variables values prior to the interruption. In addition, a *route failure timer* is employed to prevent infinite wait for RRN messages, and is started whenever a RFN is received. Upon expiration of this timer, the frozen timers of TCP are reset hence allowing the TCP congestion control to be invoked normally.

THE ELFN APPROACH

The Explicit Link Failure Notification (ELFN) [Holland 1999] is a cross-layer proposal in which TCP also interacts with the routing protocol in order to detect route failure and take appropriate actions. Here, ELFN messages are sent back to the TCP sender from the node detecting the failure. Such messages are carried by the routing protocol that needs to be adapted for this purpose. In fact, the DSR's route error message has been modified to carry a payload similar to the "host unreachable" message of the Internet Control Message

Protocol (ICMP) [Tanenbauml996]. Basically, the ELFN messages contain sender and receiver addresses and ports, as well as the TCP sequence number. This way, the modified TCP is able to distinguish losses caused by congestion from the ones due to mobility. In ELFN, whenever the TCP sender receives an ELFN message it enters a "stand-by" mode in which its timers are disabled and probe packets are sent regularly towards the destination in order to detect route restoration. Upon receiving an ACK packet, the sender leaves the "stand-by" mode and resumes transmission using its previous timer values and state variables.

FIXED RTO

The fixed RTO scheme [Dyer2001] relies on the idea that routing error recovery should be accomplished in a fast fashion by the routing algorithm. As a result, any disconnection should be treated as a transitory period which does not justify the regular exponential backoff mechanism of TCP being invoked, as this can cause unnecessarily long recovery delays. Thus, it disables such a mechanism whenever two successive retransmissions due to timeout occur, assuming that it actually indicates route failure. By doing so, it allows the TCP sender to retransmit at regular intervals instead of at increasingly exponential ones. In fact, the TCP sender doubles the RTO once and if the missing packet does not arrive before the second RTO expires, the packet is retransmitted again and again but the RTO is no longer increased. It remains fixed until the route is recovered and the retransmitted packet is acknowledged.

THE ATCP PROTOCOL:

Different from previously discussed approaches, the Ad hoc TCP (ATCP) protocol [Liu2001] does not impose changes to the standard TCP itself. Rather, it implements an intermediate layer between the network and the transport layers in order to provide an enhanced performance to TCP and still maintain interoperation with non-ATCP nodes.

More specifically, ATCP relies on the ICMP protocol and on the Explicit Congestion Notification (ECN) [ECNwww] scheme to detect/distinguish network partition and congestion, respectively. This way, the intermediate layer keeps track of the packets to and from the transport layer so that the TCP congestion control is not invoked when it is not really needed.

TCP-DOOR

Due to its dynamic environment, mobility in MANETs is extremely frequent. Therefore, a natural effect of mobility is that the packet usually arrive out-of-order (OOO) at the destination. If the OOO delivery event is appropriately monitored, it might be just enough to detect link failure inside the network and, hence, be able to effectively distinguish between mobility and congestion. The TCP-DOOR (Detection of Out-of-Order and Response) [Wang2002] protocol focuses on the idea that OOO delivery of packets can happen frequently in MANETs as a result of nodes mobility. TCP-DOOR imposes changes to TCP code but does not require intermediate nodes to cooperate, which represents its main differentiation from all previously described proposals. In this way, TCPDOOR detects OOO events and responds accordingly as explained below.

FAIRNESS-RELATED

Compared to the number of mobility-related studies, little has been done to address the serious unfairness conditions raised by TCP over adhoc networks. In this section we present some important solutions in this category. Other related studies can be found in [Bottigliengo2004].

COPAS

As we have seen before, the problem of capture is severe due to the interplay of the MAC layer and TCP backoff policies and results in a single node within its radio range being able to access the medium at all times, while others in its neighborhood starve. To the same extent as mobility related issues, the capture problem drastically affects TCP performance and is stressed in wireless MAC protocols that employ exponential backoff schemes such as IEEE 802.11 and FAMA (Floor Acquisition Multiple Access) [Fullmerl995] as their backoff mechanisms always favor the last successful station.

A protocol called COPAS (COntention-based PAth Selection) has been proposed in [Cordeiro2002] to address TCP performance drop due to the capture problem and resulting unfairness. COPAS implements two novel routing techniques in order to contention-balance the network, namely, the use of disjoint forward (for TCP data) and reverse (for TCP ACK) paths to reduce the conflicts between TCP packets traveling in opposite directions, as well as a dynamic contention-balancing technique that continuously monitors network contention and selects routes with minimum MAC layer contention.

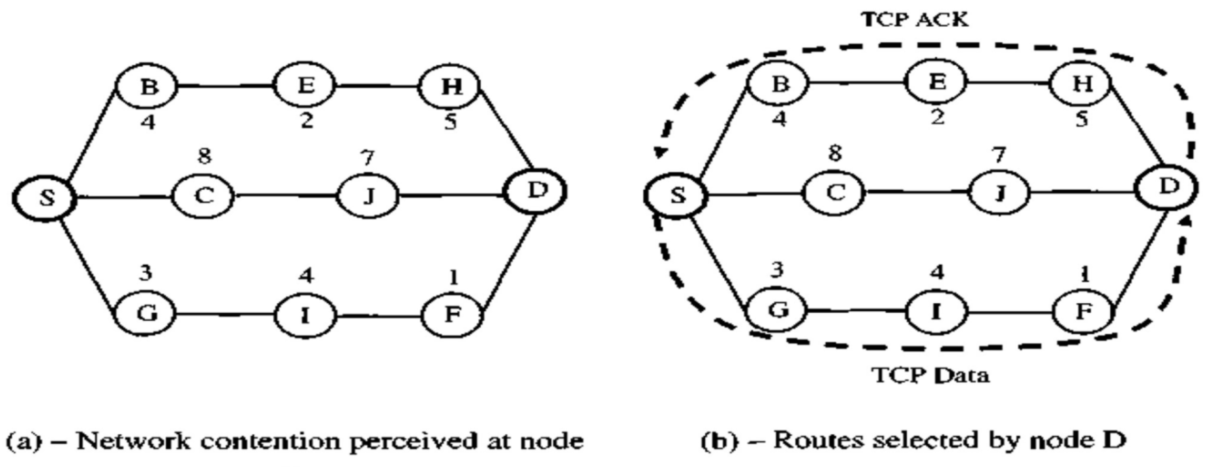

B
E
H
4
2
5
8
7
S
C
J
D
3
4
1
G
I
F
TCP ACK
TCP Data
(a) – Network contention perceived at node D
(b) – Routes selected by node D

Figure 7.9 – Route establishment in COPAS [Taken from IEEE Publication Cordeiro2002]

To illustrate this, consider the scenario of Figure 7.9(a) wherein the source node S sends a RREQ packet towards the destination node D. In this case and with the contention values as depicted in the Figure 7.9(a), 392 *AD HOC & SENSOR NETWORKS* the destination first applies the disjointness path rule and finds out routes *i* = <S-B-E-H-D>,; = < S-C-J-D >, and *k* = < S-G-I-F-D> to be disjoint. Next, it applies the minimum contention sum rule and end up selecting routes *i* and *k* to be used as reverse (for TCP ACK) and forward (for TCP data) paths respectively, as showed in Figure 7.9(b). Employing disjoint forward and reverse paths is also desirable for robustness reasons. Capture conditions can be so severe that links appear to be broken even when there is no mobility. Therefore, to guarantee continuous operation even in link breakage situations COPAS makes use of previously established forward and reverse routes. In a capture scenario, it is usually the MAC layer which reports to the network layer the link breakage since it is in this layer where the capture problem is rooted. When a route is disconnected, the immediate upstream node of the broken link sends a RERR message to the source of the route notifying the route invalidation.

UNIT – III

BASICS OF WIRELESS SENSORS AND APPLICATIONS

THE MICA MOTE:

The Mica Mote [MICAwww] shown in Figure 8.2(a) is a comprehensive sensor node developed by University of California at Berkeley for this application and marketed by Crossbow. It uses an Atmel Atmega 103 microcontroller running at 4 MHz, with a radio operating at the 916 MHz frequency band which provides bidirectional com unication at 40 kbps. A pair of AA batteries provides the required energy. The Mica Board, shown in Figure 8.2(b), is stacked to the processor board via the 51 pin extension connector. It includes temperature, photo resistor, barometer, humidity, and thermopile sensors. To conserve energy, later designs include an A/D Converter and an 8x8 power switch on the sensor board, the bypassing of the DC booster, etc. To protect sensors from the variable weather condition, the mica mote is packaged in an acrylic enclosure, which does not obstruct the sensing functionality and the radio communication. MICA 2 motes have three modes based RF frequency band, MPR400 using 915 MHz, MPR 410 employing 433 MHz and MPR420 based on 315 MHz. Three miniaturized sizes (1/4) of Mote are also available as MICA2DOT. Another version of MICA 2 using a combination of ultrasound and RF signal has been jointly developed by Crossbow and MIT and has a flexibility to configure either as a listener or a beacon transmitter. More details on Mica Motes and version 2 can be obtained from details of different types of commercially available sensor transducers could be obtained from many web sites. **Sensing and**

COMMUNICATION RANGE

A wireless sensor network (WSN) consists of a large number of sensor nodes (SNs) and exploring their best possible use is a challenging problem. As the main objective of a SN is to monitor some physical quantity in a given area, the sensors need to be deployed with adequate density so that sensing of the complete area can be done, without leaving any void or unsensed area. In other words, given the sensing range of each SN and the area to be covered, adequate number of SNs should be needed to be placed throughout the area so that no corner is left out. The SNs can be placed deterministically at pre-specified locations or could be distributed randomly. So, if N SNs are put in an area A=LxL, then the SNs density can be given by *X — NIA.* The sensing range rs of each sensor can be shown in Figure 8.3. As illustrated in Figure 8.3, each SN has its own sensing range and to cover the whole space, adjacent SNs need to be located close to each other and at the most at a distance of 2rs from each other as illustrated by, three positions of SN2, SN2 and SN2 with different overlapping areas between SN| and SN2. If the SNs are unifonnly distributed with the node density of *X,* theprobability that there are m SNs within the space of S is Poisson distributed [Bettstetter2002] as m! Where space *S — 7Cr~* for two dimensional spaces. This gives the probability that the monitored space is not covered by any SN and hence the probability pcovcr of the coverage by at least one SN is:

Paw,,=l-P(0) = l - ^

This above relation gives an idea about the coverage of the area so that adequate number of sensors could be deployed.

DESIGN ISSUES:

The advancement in technology has made it possible to have a network of 100s or even thousands of extremely small, low powered devices equipped with programmable computing, multiple parameter sensing and wireless communication capability, enhancing the reliability, accuracy of data and the coverage area. In short, some of the advantages of WSN over wired ones are as follows:

• Ease of deployment - These wireless sensors can be deployed (dropped from a plane or placed in a factory) at the site of interest without any prior organization, thus reducing the installation cost and time, and also increasing the flexibility of deployment;
• Extended range - One huge wired sensor (macro-sensor) can be replaced by many smaller wireless sensors for the same cost. Such a macro-sensor can sense only a limited region whereas a network of smaller sensors can be distributed over a wider range;
• Fault tolerant -With macro-sensors, the failure of one node makes that area completely unmonitored till it is replaced. With wireless sensors, failure of one node does not affect the network operation substantially as there are other adjacent nodes collecting similar data. At most, the accuracy of data collected may be somewhat reduced;
• Mobility - Since these wireless sensors are equipped with battery, they can possess limited mobility (e.g., if placed on robots). Thus, if a region becomes unmonitored we can have the nodes rearrange themselves to distribute evenly, i.e., these nodes can be made to move towards area of interest but having lower mobility as compared to ad hoc networks. Traditional routing protocols defined for MANETs (discussed in previous chapters) are not well suited for wireless sensor networks due to the following reasons:
• As we mentioned earlier, w centric, routing to and from specific nodes in these networks is not required;
• Adjacent nodes may have similar data. So, rather than sending data separately from each sensor node to the requesting node, it is desirable to aggregate similar data before sending it;
• The requirements of the network change with the application and hence, it is application-specific. For example, in some applications, the sensor nodes are fixed and not mobile while others may need data based only on some selected attributes (viz., attribute is fixed in this network). ireless sensor networks are "data centric", where data is requested based on particular criteria such as "which area has temperature 35°C";
• In traditional wired and wireless networks, each node is given a unique identification (e.g., an IP address) used for routing. This cannot be effectively used in sensor networks because, being data.

ENERGY CONSUMPTION:

Minimizing the energy consumption of WSs is critical yet a challenge for the design of WSNs. The energy consumption in WSN involves three different components: Sensing Unit (Sensing transducer and A/D Converter), Communication Unit (transmission and receiver radio), and Computing/Processing Unit. In order to conserve energy, we may make some SNs go to sleep mode and need to consider energy consumed in that state.
SENSING TRANSDUCER is responsible for capturing the physical parameters of the environment. Its basic function is to do physical signal sampling and convert into electrical signals. The energy consumption of this part depends on the hardware and the application and sensing energy is only a small part of the total energy consumption.
A/D CONVERTER: Based on paper [www.moteiv.com], an AD Converter for sensor consumes only 3.1 *JLlW* , in 31 *pJ/8-bit* sample at lVolt supply. The standby power consumption at IV supply is *lpW.*Assuming the D/C is not noise limited, the lower bound on energy per sample for the successive approximation architecture is roughly computed as: *Emin-C,otaNref 2*, where *Ctotai* is the total capacitance of the array including the bottom plate parasites, and *Vref* is a common mode input voltage the comparator works under.
TRANSMISSION ENERGY: Based on [Heinzelman2000] the transmission energy transmits a k-bit message to distance d can be computed as:*ETx(k,d)=ETx.eiec(k)+ETx.amp(k,d)=Eetec*k+ £ *k*d2*, where *Erx-eiec* is the transmission electronics energy consumption, *Erx.amp* is the transmit amplifier energy consumption. Their model assumes *ETx_dec=ERx,elec=Eelec=50nJ/bit*, and *£amp=\00pJ/bit/m2*

RECEIVER ENERGY: To receive a k bit message, the energy consumed is £fcW=£,ftc.efc/&J=.£'efec*£
COMPUTATION: The computing unit associated with a wireless sensor is a microcontroller/ processor with memory which can control and operate the sensing, computing and communication unit. The energy consumption of this unit has mainly two parts: switching energy and leakage energy. Switching energy is expressed as [Shih2001] *Eswuch=Ct0,aiVdd 2,* where *Cmai* is the total capacitance switched by the computation and *Vdd* is the supply voltage. Dynamic voltage scaling (DVS) scheme is used to adaptively adjust operating voltage and frequency meet the dynamically changing workload without degrading performance thus saving energy. Leakage energy is the energy consumed when no computation work is done. It can be expressed as: *Eleakageup* = (V*'ddi)l\enVr* , where *VT* is the thermal voltage, *n'* and *I0* are the parameters of processor and can get from experiment. For StrongARM SA-1100, *n'* =21.26 and /0= 1.196mA

CLUSTERING OF SENSORS:

From the preceding sections, it is clear that enough number of SNs need to be deployed if every corner of the area of interest; need to sensed for continuous monitoring and necessary action. In addition, successful transfer of sensed data to adjacent SNs necessitates minimum communication distance covered by the wireless radio to be at least twice that of sensing range. So, sensing distance and communication coverage are co-related. It is widely accepted that the energy consumed in one bit of data can be used to perform a large number of arithmetic operations in the sensor processor (power consumed in 1-bit transfer to 100m equals 3000 instructions [Pottie2000]). Moreover, the way sensors are deployed, physical environment would produce similar in close-by SNs and transmitting such data could be termed as more or less redundant.

Therefore, all these facts encourage using some kind of grouping (clustering) so that data from SNs belonging to a single cluster can be combined together in an intelligent way (aggregation) using local transmissions to transfer only the compact data. This can not only reduce the global data to be transferred and localized most traffic to within each individual cluster, but also reduces the traffic and hence contention in a WSN. A lot of research gone into testing coverage of areas by k-sensors (k>l) [Choi2004, Liu2004, Megerian2005, Zhang2005] clustering adjacent SNs and defining the size of the cluster [Chowdhury2005] so that the cluster heads (CHs) can easily get data from their own cluster members and CHs can also communicate among themselves and exchange data. If each cluster is covered by more than one subset of SNs all the time, then some of the SNs can be put into sleep mode so as to conserve energy while keeping full coverage of each cluster and the area. The use of a second smaller radio has been suggested for waking up the sleeping sensor [Miller2005], thereby conserving the power of main wireless transmitter. In this section, we consider cluster formation in two types of WSNs, one place in a predetermined grid form and another, when SNs are placed randomly within a given area.

APPLICATIONS

Thousands of sensors over strategic locations are used in a structure such as an automobile or an airplane, so that conditions can be constantly monitored both from the inside and the outside and a real-time warning can be issued whenever a major problem is forthcoming in the monitored entity. These wired sensors are large (and expensive) to cover as much area is desirable. Each of these need a continuous power supply and communicates their data to the end-user using a wired network. The organization of such a network should be pre-planned to find strategic position to place these nodes and then should be installed appropriately. The failure of a single node might bring down the whole network or leave that region completely un-monitored. Un-attendability and some degree of fault tolerance in these networks are especially desirable in those applications where the sensors may be embedded in the structure or places in an inhospitable terrain and could be inaccessible for any service. Undoubtedly, wireless sensor networks have been conceived with military applications in mind, including battlefield surveillance and tracking of enemy activities. However, civil applications considerably outnumber the military ones and are applicable to many practical situations

CLASSIFICATIONS OF WSNS:

A WSN is deployed primarily to collect sensed data by different WSs and it is critical to see how requently the sensed values are collected. Looking at various ways in which one can employ the network resources, WSNs can be classified on the basis of their mode of operation or functionality, and the type of target applications. Accordingly, we hereby classify WSNs into three types:

• **Proactive Networks -** The nodes in this network periodically switch on their sensors and transmitters, Sense the environment and transmit the data of interest. Thus, they provide a snapshot of the relevant Parameters at regular intervals and are well suited for applications requiring periodic data monitoring.

• **Reactive Networks -** In this scheme, the nodes react immediately to sudden and drastic changes in the value of a sensed attribute. As such, these are well suited for time critical applications.

• **Hybrid Networks -** This is a combination of both proactive and reactive networks where sensor node not only send data periodically, but also respond to sudden changes in attribute values.

Once the type of network is decided, protocols that efficiently route data from the SNs to the users have to be designed, perhaps using a suitable MAC protocol to avoid collisions and subsequent energy consumption. Attempts should be made to distribute energy dissipation evenly among all nodes in the network, as it is usually not common to assume the presence of specialized high-energy nodes in the network. In this chapter we cover proactive, reactive and hybrid protocols, while highlighting the fact that the protocols ought to be directly related to application requirements. **Architecture of Sensor Networks:**

Due to the principle differences in application scenarios and underlying communication technology, the architecture of WSNs will be drastically different both with respect to a single WS and the network as a whole. The typical hardware platform of a wireless sensor node will consist of:

• Simple embedded microcontrollers, such as the Atmel or the Texas Instruments MSP 430. A decisive characteristic here is, apart from the critical power consumption, an answer to the important question whether and how these microcontrollers can be put into various operational and sleep modes, how many of these sleep modes exist, how long it takes and how much energy it costs to switch between these modes. Also, the required chip size and computational power and on-chip memory are important

• Currently used radio transceivers include the RFM TR1001 or Infineon or Chip on devices; similar radio modems are available from various manufacturers. Typically, ASK or FSK is used, while the Berkeley Pico Nodes employ OOK modulation. Radio concepts like ultra-wideband are in an advanced stage (e.g., the projects undertaken by the IEEE 802.15 working group). A crucial step forward would be the introduction of a reasonably working wake-up radio concept, which could either wake up all SNs in the vicinity of a sender or even only some directly addressed nodes. A wake-up radio allows a SN to sleep and to be wakened up by suitable transmissions from other nodes, using only a low-power detection circuit. Transmission media other than radio communication are also considered, e.g., optical communication or ultra-sound for underwater-applications. However, this largely depends on the application

• Batteries provide the required energy. An important concern is battery management and whether and how energy scavenging can be done to recharge batteries in the field. Also, self-discharge rates, self recharge rates and lifetime of batteries can be an issue, depending on the application;

• The operating system and the run-time environment is a hotly debated issue in the literature. On one hand, minimal memory footprint and execution overhead are required while on the other, flexible means of combining protocol building blocks are necessary, as meta information has to be used in many places in a protocol stack (e.g., information about location, received signal strength, etc., has an influence on many different protocol functions). Consequently, we believe that structures like blackboards, publish/subscribe or tuple spaces are an interesting starting point for the run-time environments for such SNs.

MAC LAYER:

The MAC and the routing layers are the most active research areas in WSNs. Therefore, an exhaustive discussion of all schemes is impossible. However, most of the existing work addresses how to make SNs sleep as long as possible. Consequently, these proposals often tend to include at least some aspects of TDMA. The wireless channel is primarily a broadcast medium. All nodes within radio range of a node can hear its transmission. This can be used as a uni cast medium by specifically addressing a particular node and all other nodes can drop the packet they receive. There are two types of schemes available to allocate a single broadcast channel among competing nodes: Static Channel Allocation and Dynamic Channel Allocation.

• Static Channel Allocation: In this category of protocols, if there are N SNs, the bandwidth is divided into N equal portions in frequency (FDMA), in time (TDMA), in code (CDMA), in space (SDMA) or In schemes such as OFDM or are only a small and fixed number of SNs, each of which has Buffered (heavy) load of data

• Dynamic Channel Allocation: In this category of protocols, there is no fixed assignment of bandwidth. When the number of active SNs changes dynamically and data becomes burst at arbitrary SNs, it is most advisable to use dynamic channel allocation scheme. These are contention-based schemes, where SNs contend for the channel when they have data while minimizing collisions with other SNs transmissions. When there is a collision, the SNs are forced to retransmit data, thus leading to increased wastage of energy and unbounded delay.

As we will see shortly, in a hierarchical clustering model, once clusters have been formed, it is desirable to keep the number of nodes in the cluster fixed and due to hierarchical clustering; the number of nodes per cluster is not kept large. So, it may be better to use one of the static channel allocation schemes. In his scheme, each node transmits data in its own slot to the cluster head and at all other times, its radio can be switched off, thereby saving valuable energy.

When it is not feasible to use TDMA, the n des can use non persistent CSMA since the data packets are of fixed size.TDMA is suitable for either proactive or reactive type of networks. In proactive networks, as we have the nodes transmitting periodically, we can assign each node a slot and thus avoid collisions. In reactive Networks, since adjacent nodes have similar data, when a sudden change takes place in some attribute being sensed, all the nodes will respond immediately. This will lead to collisions and it is possible that the data may never reach the user on time. For this reason, TDMA is employed so that each node is given a slot and they transmit only in that slot. Even though this increases the delay and many slots might be empty, it is better than the energy consumption incurred due to dynamic channel allocation schemes.

Design Issues: As with MAC protocols for traditional MANETs, WSNs have their own inherent characteristics that need to be addressed. Below we discuss some of the most important ones involved in the design of MAC protocols for WSNs.

Coping up with Node Failure: When many SNs have failed, the MAC and routing protocols must accommodate formation of new links and routes to other SNs and the BS. This may require dynamically adjusting transmit powers and signaling rates on the existing links, or rerouting packets through regions of the network with higher energy level.

Sources of Resource Consumption at the MAC Layer: There are several aspects of a traditional MAC protocol that have negative impact on wireless sensor networks including:

• Collisions - When a transmitted packet is corrupted due to a collision, it has to be discarded. The follow-on retransmission increases the energy consumption and hence increases the latency

• Overhearing - SNs listen to transmissions that are destined to other SNs

• Control packets overhead - Sending and receiving control packets consume energy and reduce the payload. This overhead increases linearly with node density. Moreover, as more SNs fail in the network, more control messages are required to self-configure the system, resulting in more energy consumption

• Idle Listening -Waiting to receive anticipated traffic that is never sent. This is especially true in many sensor network applications. If nothing is sensed, SNs are in the idle mode for most of the time.

Measures to Reduce Energy Consumption: One of the most cited methods to conserve energy in sensor networks is to avoid listening to idle channels, that is, neighboring nodes periodically sleep (radio off) and auto synchronize as per sleep schedule. It is important to note that fairness, latency, throughput and bandwidth utilization are secondary in the WSNs.

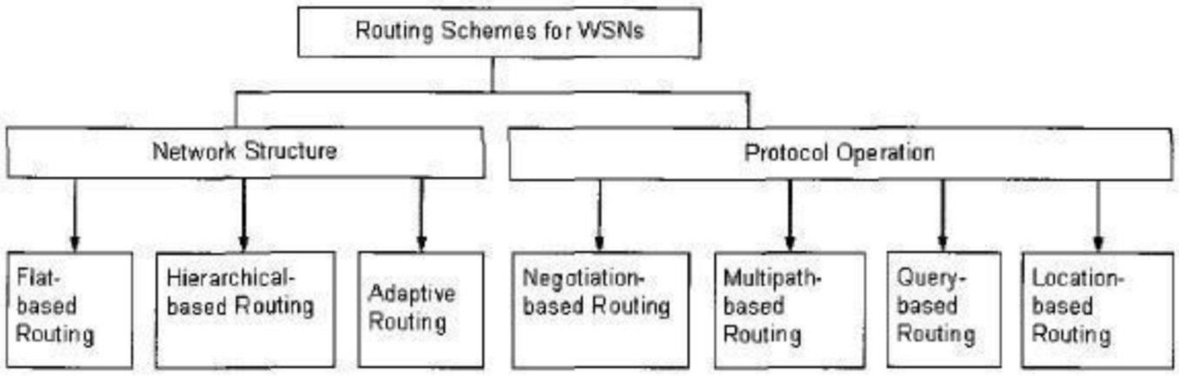
Comparison of Scheduling & Reservation-based andContention-based MAC Design: One approach of MAC design for WSNs is based on reservation and scheduling, for example TDMA-based protocols that conserve more energy as compared to contention-based protocols like the IEEE 802.11 DCF. This is because the duty cycle of the radio is increased and there is no contention-introduced overhead and collisions. However, formation of cluster, management of inter-cluster communication, and dynamic adaptation of the TDMA protocol to variation in the number of nodes in the cluster in terms of its frame length and time slot assignment are still the key challenges.

MAC Protocols: WSNs are designed to operate for long time as it is rather impractical to replenish the batteries. However, nodes are in idle state for most time when no sensing occurs. Measurements have shown that a typical radio consumes the similar level of energy in idle mode as in receiving mode. Therefore, it is important that nodes are able to operate in low duty cycles. As far as the MAC layer is concerned, some of the more recent and relevant studies in this area are Pico Radio MAC, the S-MAC, the SMACS and the STEM As many of these protocols share common characteristics; in this section we discuss only those which are most prominent and necessary to understand the others. It is also important to note that more traditional MAC schemes such as FDMA, TDMA, CDMA, SDMA and a combination of these can also be employed. However, as these techniques are widely known, we do not discuss them.

ROUTING LAYER:

By now, it must be clear that WSNs differ from traditional wireless networks. Conventional flooding-based protocols widely employed in MANETs suffer from data explosion problem, i.e., if a node is a common neighbor to nodes holding the same data item, then it will get multiple copies of the same data item. Therefore, the protocol wastes resources by sending and receiving duplicate data copies. In addition, flooding does not scale well in large networks and wastes resources.

The goal is to send the data from source node(s) to a known destination node, i.e., the BS. The destination node or the sink node is known and addressed by means of its location. A BS may be fixed or mobile, and is capable of connecting the sensor network to an existing infrastructure (e.g., Internet) where the user can have access to the collected data. The task of finding and maintaining routes WSNs is nontrivial since energy restrictions and sudden changes in node status (e.g., failure) cause frequent unpredictable topological changes. Thus, the main objective of routing techniques is to minimize the energy consumption in order to prolong WSN lifetime. To achieve this objective, routing protocols proposed in the literature employ some well-known routing techniques as well as tactics special to WSNs. To preserve energy, strategies like data aggregation and in-network processing, clustering, different node role assignment, and data-centric methods are employed. In sensor networks, conservation of energy is considered



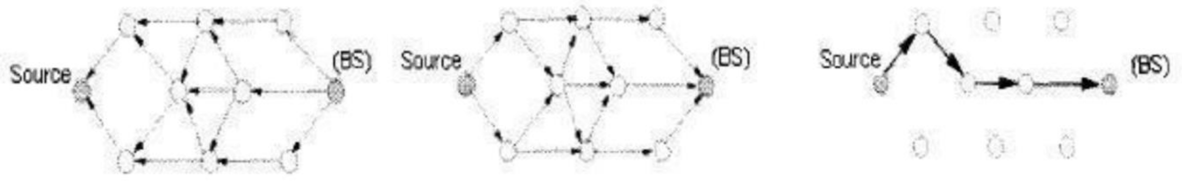
Classification of routing protocols for WSNs

In sensor networks, conservation of energy is considered relatively more important than quality of data sent. Therefore, energy-aware routing protocols need to satisfy this requirement. Routing protocols for WSNs have been extensively studied in the last few years. Routing protocols for WSNs can be broadly classified into flat-based, hierarchical-based, and adaptive, depending on the network structure.

In Flat-based routing, all nodes are assigned equal role. In hierarchical based routing, however, nodes play different roles and certain nodes, called cluster heads (CHs), are given more responsibility. In adaptive routing, certain system parameters are controlled in order to adapt to the current network conditions and available energy levels. Furthermore, these protocols can be classified into multipath-based, query-based, negotiation-based, or location-based routing techniques. In this section we use a classification according to the network structure and protocol operation (i.e., routing criteria), and is shown in Figure 9.8. In majority of applications, sensor nodes are expected to be stationary. Thus, it may be preferable to have table driven routing protocols rather than employing reactive schemes where a significant amount of energy is used in route discovery and setup. Another class of routing protocols is called the cooperative routing protocols. Where in SNs send data to a CH where data can be aggregated and may be subjected to further processing, hence reducing route cost in terms of energy use. Several other protocols, in turn, rely on timing and position information and are also covered in this section.

- **Network Structure Based** In this class of routing protocols, the network structure is one of the determinant factors. In addition, the network structure can be further subdivided into flat, hierarchical and adaptive depending upon its organization.
- **Flat Routing** In flat routing based protocols, all nodes play the same role. Here, we present the most prominent protocols falling in this category.
- **Directed Diffusion** Directed Diffusion [Intanagonwiwat2000] is a data aggregation and dissemination paradigm for sensor networks. It is a data-centric (DC) and application-aware approach in the sense that all data generated by sensor nodes is named by attribute-value pairs. Directed Diffusion is very useful for applications requiring dissemination and processing of queries. The main idea of the DC paradigm is to combine the data coming from different sources en-route (in-network aggregation) by eliminating redundancy, minimizing the number of transmissions; thus saving network energy and prolonging its lifetime. Unlike traditional end-to-end routing, DC routing finds routes from multiple sources to a single destination (BS) that allows in-network consolidation of redundant data. In Directed Diffusion, sensors measure events and create gradients of information in their respective neighborhoods. The BS requests data by broadcasting *interests*, which describes a task to be done by the network.

Interest diffuses through the network hop-by-hop, and is broadcast by each node to its neighbors. As the interest is propagated throughout the network, gradients are setup to draw data satisfying the query towards the requesting node. Each SN that receives the interest setup a gradient toward the SNs from which it receives the interest. This process continues until gradients are setup from the sources back to the BS. The strength of the gradient may be different towards different neighbors, resulting in variable amounts of information flow. At this point, loops are not checked, but are removed at a later stage. Figure 9.9 depicts an example of the operation of directed diffusion. Figure 9.9(a) presents the propagation of interests, Figure 9.9(b) shows the gradients construction, and Figure 9.9(c) depicts the data dissemination. When interests fit gradients, paths of information flow are formed from multiple paths, and the best paths are reinforced so as to prevent further flooding according to a local rule. In order to reduce ommunication costs, data is aggregated on the way. The BS periodically refreshes and re-sends the interest when it starts to receive data from the source(s). This retransmission of interests is needed because the medium is Inherently unreliable.



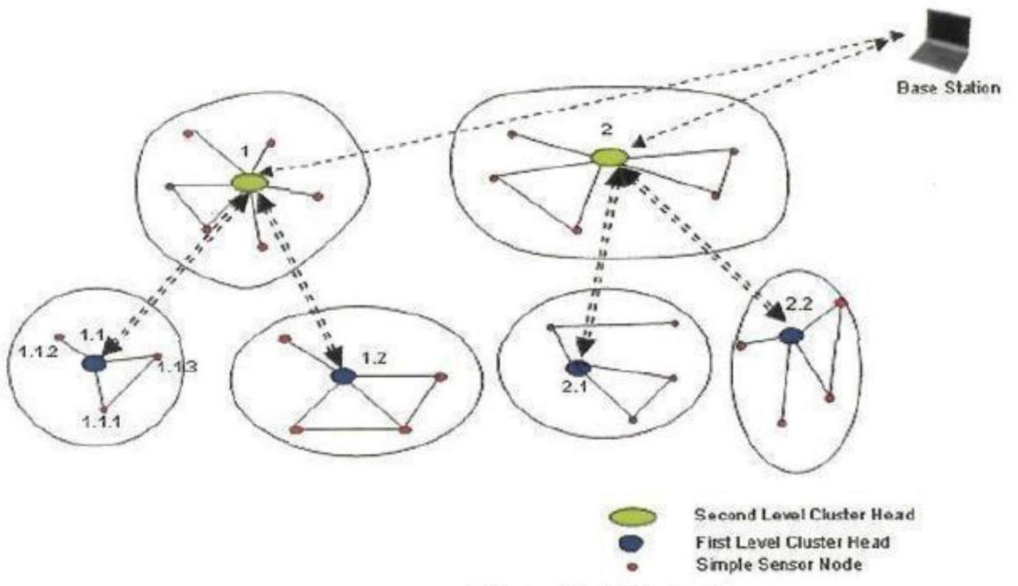
(a) Interest Propagation (b) Gradients set up (c) Data transmission and path reinforcement

Sensor nodes in a directed diffusion-based network are application aware, which enables diffusion to achieve energy savings by choosing empirically good paths and by caching and processing data in the network. An application of directed diffusion is to spontaneously propagate an important event to regions of the sensor network. Such type of information retrieval is well suited for persistent queries where requesting nodes expect data that satisfy a query for a period of time. However, it may be unsuitable for historical or one-time queries as it is not worth setting up gradients which employ the path only once. The performance of data aggregation methods employed by the directed diffusion paradigm is affected by the location of the source nodes in the network, the number of sources, and the network topology. In order to investigate these factors, two models of source placement shown in Figure 9.10 have been investigated. These models are called the event radius (ER) model, and the random sources (RS) model. In the ER model, a single point in the network area is defined as the location of an event. For example, this may correspond to a vehicle or some other phenomenon being tracked by the sensor nodes. All nodes within a distance S (called the sensing range) of this event that are not sinks, are considered to be data sources. In the RS model, K nodes that are not sinks are randomly selected to be sources. Unlike the ER model, in the RS model the sources are not necessarily closed to each other. In both of these source placement models, however, for a given energy budget, a greater number of sources can be connected to the sink. Therefore, the energy savings with aggregation in directed diffusion can be transformed to provide a greater degree of robustness as per dynamics of the sensed activity.

Sequential Assignment Routing (SAR) The routing scheme in SAR [Sohrabi2000] is dependent on three factors: energy resources, QoS on each path, and the priority level of each packet. To avoid single route failure, a multi-path approach coupled with a localized path restoration scheme is employed. To create multiple paths from a source node, a tree rooted at the source node to the destination nodes (i.e., the set of BSs) is constructed. The paths of the tree are defined by avoiding nodes with low energy or QoS guarantees. At the end of this process, each sensor node is part of multi-path tree. For each SN, two metrics are associated with each path: delay (which is an additive QoS metric); and energy usage for routing on that path. The energy is measured with respect to how many packets will traverse that path. SAR calculates a weighted QoS metric as the product of the additive QoS metric and a weight coefficient associated with the priority level of the packet. The goal of SAR is to minimize the average weighted QoS metric throughout the lifetime of the network. Also, a path recomputation is carried out if the topology changes due to node failures. As a preventive measure, a periodic re-computation of paths is triggered by the BS to account for any changes in the topology. In addition, a handshake procedure based on a local path restoration scheme between neighboring nodes is used to recover from a failure.

Hierarchical Routing

Hierarchical, or cluster-based, routing has its roots in wired networks, where the main goals are to achieve scalable and efficient communication. As such, the concept of hierarchical routing has also been employed in WSN to perform energy-efficient routing. In a hierarchical architecture, higher energy nodes (usually called cluster heads) can be used to process and send the accumulated information while low energy nodes can be used to sense in the neighborhood of the target and pass on to the CH. In these cluster-based architectures, creation of clusters and appropriate assignment of special tasks to CHs can contribute to overall system scalability, lifetime, and energy efficiency. An example of a general hierarchical clustering scheme is depicted in Figure 9.11. As we can see from this figure, each cluster has a CH which collects data from its cluster members, aggregates it and sends it to the BS or an upper level CH. For example in Figure 9.11, nodes 1.1.1, 1.1.2, 1.1.3 and 1.1 form a cluster with node 1.1 as the CH. Similarly, there exist other CHs such as 1.2, etc. These CHs, in turn, form a cluster with node 1 as their CH. So, node 1 becomes a second level CH as well. This pattern is repeated to form a hierarchy of clusters with the uppermost level cluster nodes reporting directly to the BS. The BS forms the root of this hierarchy and supervises the entire network.



Hierarchical Clustering

A simple cluster based routing protocol (CBRP) has been proposed in [Jiang 1998]. It divides the network nodes into a number of overlapping or disjoint two-hop-diameter clusters in a distributed manner. Here, the cluster members just send the data to the CH, and the CH is responsible for routing the data to the destination. The major drawback with CBRP is that it requires a lot of hello messages to form and maintain the clusters, and thus may not be suitable for WSN. Given that sensor nodes are stationary in most of the applications this is a considerable and unnecessary overhead.

SCALABLE COORDINATION In [Estrinl999], a hierarchical clustering method is discussed, with emphasis on localized behavior and the need for asymmetric communication and energy conservation in a sensor network. In this method the cluster formation appears to require considerable amount of energy (no experimental results are available) as periodic advertisements are needed to form the hierarchy. Also, any changes in the network conditions or sensor energy level result in re-clustering which may be not quite acceptable as some parameters tend to change dynamically.

LOW-ENERGY ADAPTIVE CLUSTERING HIERARCHY (LEACH)

LEACH is introduced in [Heinzelman2000b] as a hierarchical clustering algorithm for sensor networks, called Low-Energy Adaptive Clustering Hierarchy (LEACH). LEACH is a good approximation of aproactive network protocol, with some minor differences which includes a distributed cluster formation algorithm. LEACH randomly selects a few sensor nodes as CHs and rotates this role amongst the cluster members so as to evenly distribute the energy dissipation across the cluster. In LEACH, the CH nodes compress data arriving from nodes that belong to the respective cluster, and send an aggregated packet to the BS in order to reduce the amount of information that must be transmitted. LEACH uses a TDMA and CDMA MAC to reduce intra-cluster and inter-cluster collisions, respectively. However, data collection is centralized and is performed periodically. Therefore, this protocol is better appropriate when there is a need for constant monitoring by the sensor network. On the other hand, a user may not need all the dataimmediately.

Hence, periodic data transmissions may become unnecessary as they may drain the limited energy of the sensor nodes. After a given interval of time, a randomized rotation of the role of the CH is conducted so that uniform energy dissipation in the sensor network is obtained. Based on simulation, it has been found that only 5% of the nodes actually need to act as CHs. The operation of LEACH is separated into two phases, the setup phase and the steady state phase. In the setup phase, the clusters are organized and CHs are selected. In the steady state phase, actual data transfer to the BS takes place. Clearly, the duration of the steady state phase is longer than the duration of the setup phase in order to minimize overhead. During the setup phase, a predetermined fraction of nodes, say *p,* elect themselves as CHs as follows. A sensor node chooses a random number, say r, between 0 and 1.

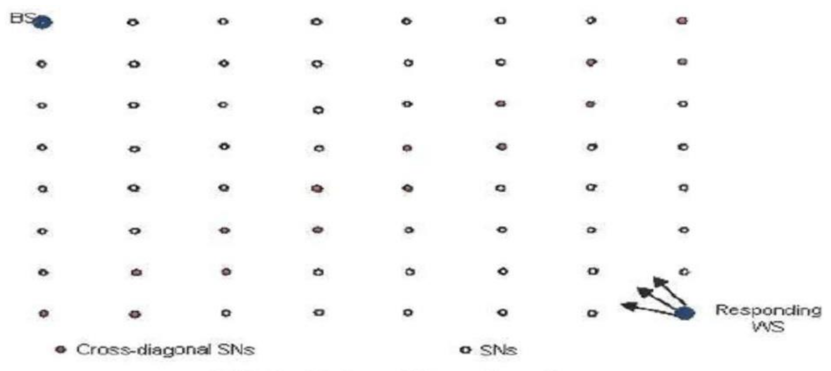
If this random number is less than a threshold value, say *T(n),* the node becomes a CH for the current round. The threshold value, in turn, is calculated based on an equation that incorporates the desired percentage to become a CH, the current round, and the set of nodes that have not been selected as a CH in the last
*{lip)*rounds, denoted by *G.* As a result, *T(n)* is given by: *T(n) = - ifneG I- p(rmod(l/ p))*

where G is the set of nodes that are involved in the CH election. Each elected CH broadcast an advertisement message to the rest of the nodes in the network, informing that they are the new CHs. All the non- CH nodes, after receiving this advertisement, decide on the cluster to which they want to attach to. In LEACH, this decision is based on the signal strength of the advertisement. The non-CH nodes then inform the corresponding CH of their decision to be a member of its cluster. Based on the number of nodes in the cluster, the CH node creates a TDMA schedule and assigns each node a time slot within this schedule where it can transmit. This schedule is then broadcast to all cluster members. During the steady state phase, sensor nodes begin sensing and transmitting data to their respective CHs. Once the CH receives the data from all of its members, it aggregates before relaying data to the BS. After a period time, which is determined a priori, the network goes back into the setup phase and initiates another round for selecting new CHs. Although LEACH is able to increase the network lifetime, there are still a number of issues regarding many assumptions. For example, LEACH assumes that all nodes can transmit with enough power to reach the BS if needed, and that every node has enough computational power to support different MAC protocols. It also assumes that nodes always have data to send, and nodes located close to each other have correlated data. Also, it is not obvious how the number of the predetermined CHs (p) is going to be uniformly distributed through the network. Therefore, there is the possibility that the elected CHs be concentrated in one part of the network. Thus, some nodes will not at all find CHs in their proximity. Finally, the protocol assumes that all nodes begin with the same amount of energy capacity in each election round. LEACH could be extended to account for non-uniform energy nodes, i.e., to use energy-based threshold. An extension to LEACH, also known as LEACH with negotiation, has been introduced in [Heinzelman2000b] with the goal of preceding data transfers with negotiations similar to meta-data descriptors used in the SPIN protocol discussed later. This ensures that only data that actually provides new information is transmitted to the CHs.

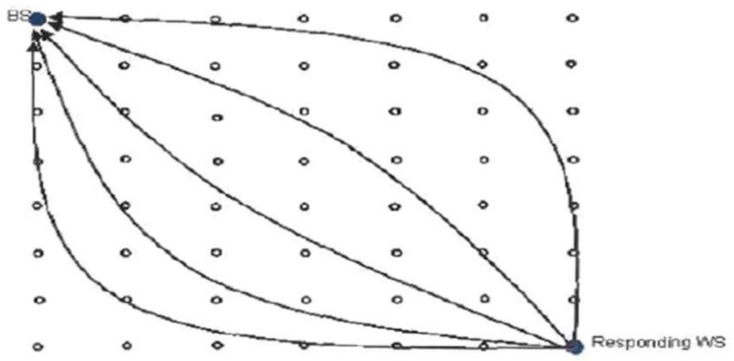
MULTIPATH-BASED ROUTING

Network performance, and possibly lifetime, in WSNs can be significantly improved if the routing rotocol is able to maintain multiple, instead of a single, paths to a destination, and protocols in this class are called multipath protocols. By employing multipath protocols, the fault tolerance (resilience) of the network is considerably increased. The fault tolerance of a protocol is measured by the likelihood that an alternate path exists between a source and a destination when the primary path fails. Clearly, this can be increased if we maintain multiple paths between the source and the destination at the expense of an increased energy consumption and traffic generation (i.e., overhead), as alternate paths are kept alive by sending periodic messages. We would also like to note here that multipath routes between a source and a destination can be or not node-disjoint. Multiple paths between a source and destination are said to be node-disjoint when there is no node overlap amongst them. For the purpose of our discussion here, we refer to

alternate routes as not being node disjoint, i.e., their routes are partially overlapped. In addition, unless otherwise noted, all multiple paths are of alternate types in this section. A set of suboptimal paths can be occasionally employed as to increases the network lifetime [Rahul2002]. These suboptimal paths are chosen by means of a probability which depends on how low the energy consumption of each path is. Packets are routed through a path with largest residual energy in the algorithm proposed in [Chang2000]. Here, the path is changed whenever a better path is discovered. The primary path is used until its energy falls below the energy of the backup path, at which time the backup path is used. By employing this mechanism, the nodes in the primary path do not deplete their energy resources through continual use of the same route, hence prolonging their lifetime. One issue with this scheme is the cost associated with switching paths, and how to deal with packets which are en-route. As we have seen before, there is a tradeoff between minimizing the total power consumed and the residual energy of a network. As a result, routing packets through paths with largest residual energy may turn out to be very energy-expensive. To minimize this effect, it has been proposed in [Li2001b] a scheme in which the residual energy of the route is relaxed a bit in order to select a more energy efficient path. Multipath routing was employed in [Dulman2003] to enhance the reliability of WSNs. This scheme is useful for delivering data in unreliable environments. Here, reliability is enhanced by providing several paths from source to destination and by sending the same packet thorough each and every path. Obviously, by using this technique, traffic increases significantly. Therefore, there is a tradeoff between the amount of traffic and the network reliability. This tradeoff is investigated in [Dulman2003] using a redundancy function that is dependent on the multipath degree and on failing probabilities of the available paths.



(a) Routing Region and Cross-diagonal SNs



(b) Paths Through cross-diagonal SNs

Determination of Multiple Paths

Directed Diffusion (discussed earlier). Based on the directed diffusion paradigm, a multipath routing scheme that finds several partially disjoint paths is presented in [Ganesan2001j. The idea is that the use of multipath routing provides a viable alternative for energy efficient recovery from failures in WSN. The motivation of using these braided paths is to keep the cost of maintaining the multiple paths low. In this scheme, the costs of alternate paths are comparable to the primary path as they tend to be much closer to the primary path.

HIGH-LEVEL APPLICATION LAYER SUPPORT:

The protocols we have presented so far are also found, albeit in some different form in traditional wired, cellular, or ad hoc networks. For specific applications, a higher level of abstraction specifically tailored to WSN appears to be useful. In this section, we outline some of the activities in this direction.

Distributed Query Processing:

The number of messages generated in distributed query processing is several magnitudes less than in centralized scheme. [Bonnet2000, Bonnet2001], discusses the application of distributed query execution techniques to improve communication efficiency in sensor and device networks. They discuss two approaches for processing sensor queries: warehousing and distributed. In the warehousing approach, data is extracted in a pre-defined manner and stored in a central database (e.g., the BS). Subsequently, query processing takes place on the BS. In the distributed approach, only relevant data is extracted from the sensor network, when and where it is needed. A language similar to the Structured Query Language (SQL) has been proposed in [Madden2003] for query processing in homogeneous sensor networks. They have developed a suite of techniques for power-based query optimization and built a prototype instantiation for the same, called TinyDB, which runs on Berkeley mica motes [MICAwww].

Sensor Databases: One can view the wireless sensor network as a comprehensive distributed database and interact with it via database queries. This approach solves, en passant, the entire problem of service definition and interfaces to WSNs by mandating, for example, SQL queries as the interface. The problems encountered here are in finding energyefficiency ways of executing such queries and of defining proper query languages that can express the full richness of WSNs. The TinyDB project [TinyDBwww] carried out at the University of California at Berkeley is looking at these issues. A model for sensor database systems known as COUGAR [COUGARwww] defines appropriate user and internal representation of queries. The sensor queries are also considered so that it is easier to aggregate the data and to combine two or more queries. In COUGAR, routing of queries is not handled. COUGAR has three-tier architecture.

• The Query Proxy: A small database component running on the sensor nodes to interpret and execute queries
• A Front end Component: A query-proxy that allows the sensor network to connect to the outside world. Each front-end includes a full-fledged database server
• A Graphical User Interface (GUI): Through the GUI, users can pose ad hoc and long running queries on the WSN. A map that allows the user to query by region and visualize the topology of sensors in the network.

Distributed Algorithms: WSNs are not only concerned with merely *sensing* the environment but also with interacting with the environment. Once actuators like valves are added to WSNs, the question of distributed algorithms becomes inevitable. One showcase is the question of distributed consensus, where several actuators have to reach a joint decision (a functionality which is also required for distributed software update, for example). This problem has been investigated to some degree for ad hoc networks [Malpani2000, Nakano2002, Srinivasan2003, Walter2001], but it has not been fully addressed in the context of WSNs where new scalability and reliability issues emerge and where the integration in the underlying, possibly data-centric routing architecture, has not yet been investigated.

Adapting to the Inherent Dynamic Nature of WSNs: Some important goals that current research in this area is aiming to achieve are as follows:

• Exploit spatial diversity and density of sensor/actuator nodes to build an adaptive node sleep schedule
• Spontaneously create and assemble network, dynamically adapt to device failure and degradation, manage mobility of sensor nodes and react to changes in task and sensor requirements
• Adaptability to drastic changes in the traffic
• Having finer control over the precision and coverage.

The Scalable Coordination Architectures for Deeply Distributed Systems (SCADDS) project SCADDS www], also a part of DARPA Sens IT program [Sens IT www], focuses on adaptive fidelity, dynamically adjusting the overall fidelity of sensing in response to task dynamics (turn on more sensors when a threat is perceived). They use additional sensors (redundancy) to extend lifetime. Neighboring nodes are free to talk to each other irrespective of their listening schedules; there is no clustering and no inter-cluster communication and interference. Adaptive Self-Configuring Ensor Network Topologies (Ascent) [Cerpa2001b], which is part of SCADDS, focuses on how to decide which nodes should join the routing infrastructure to adapt to a wide variety of environmental dynamics and terrain conditions producing regions with non uniform communication density. A node signals and reduces its duty cycle when it detects high message loss, requesting additional nodes in the region to join the network in order to relay messages to it. It probes the local communication environment and does not join the multi-hop routing infrastructure until it is helpful to do so. In addition, it avoids transmitting dynamic state information repeatedly across the network.

UNIT-IV

SECURITY

INTRODUCTION

As we have seen in the previous chapters, the advent of ad hoc networks brought with it a flurry of research primarily focused on communication and protocols in every layer of the protocol stack.Practical applications of this research range from simple chat programs to shared whiteboards and other collaborative schemes. Although intended for diverse audiences and contexts, many of these applications Share a common characteristic: they are information-centric. The information transferred may be a trivial conversation between friends, confidential meeting notes shared among corporate executives, or mission- critical military information. Despite the deployment of information-driven applications such as these, the call for ad hoc and sensor network security remains largely unanswered. Ad hoc security is not, however, a concern that has slipped through the cracks unnoticed: numerous research initiatives have been launched to surmount the challenge. Despite that, as we shall see, many questions remain open as many of the existing approaches have limited functionalities, unrealistic computational requirements, or inability to address core security issues.

Security in ad hoc networks is an essential component for basic network functions like packet forwarding and routing and network operation can be easily jeopardized if countermeasures are not embedded into the basic network functions at the early stages of their design. In ad hoc and sensor networks, the basic functions are carried out by all available nodes. This very difference is at the core of the security problems that are specific to these networks. If *a* priori trust relationship exists between the nodes of an ad hoc network, entity authentication can be sufficient to assure the correct execution of critical network functions. A priori trust can only exist in a few special scenarios like military networks and corporate networks, where a common, trusted authority manages the network, and requires tamper-proof hardware for the implementation of critical functions. An environment where a common, trusted authority exists is called a *managed environment.* On the other hand, entity authentication in a large network raises key management requirements. When tamper-proof hardware and strong authentication infrastructure are not available, like for example in an *open environment* where a common authority that regulates the network does not exist, any node of an ad hoc network can endanger the reliability of basic network functions. The correct operation of the network also requires fair share of the functions by each participating node as power saving is a major concern. The considered threats are thus not just limited to maliciousness, and a new type of misbehavior called selfishness should also be taken into account to prevent nodes that simply do not cooperate.

With the *lack of a priori trust*, a classical network security mechanism based on authentication and access control cannot cope up with selfishness. Cooperative security schemes seem to offer the only reasonable solution wherein node misbehavior can be detected through the collaboration between a numbers of nodes, assuming that a majority of nodes do not misbehave. Therefore, security in ad hoc networks is a much harder task than in traditional wired networks and we need to first understand the differences between security in centralized and distributed systems. Then, we delve into the specifics of security over ad hoc and sensor networks including key management schemes, secure routing algorithms, cooperation, and intrusion detection systems.

DISTRIBUTED SYSTEMS SECURITY According to the most widely used categorization of threats to computer systems, the threats are divided into three categories [Zwicky2000]: disclosure threats, integrity threats and denial of service threats. This by no means covers all the possible threats, but will suffice for our purposes. The disclosure threat involves the leakage of information from the system to a party that should not have seen the information and is a threat against the confidentiality of the information. The integrity threat involves an unauthorized modification of information. Finally, the denial of service threat involves inability to access a system resource that is being blocked by a malicious attacker. **Security in Ad Hoc Networks** As we know, there is no fixed infrastructure in ad hoc networks and as the name implies they are formed on the fly. The devices connect to each other in their own communication range via wireless links. Individual devices act as routers when relaying messages to other distant devices. The topology of an ad hoc network is not fixed either. It changes all the time when these mobile stations move in and out of each others transmission range. All this makes ad hoc networks very vulnerable to attacks and the security issues become very complex. In this section we give an overview of the security issues over ad hoc and sensor networks.

<u>REQUIREMENTS</u>

The security services of ad hoc networks are not altogether different than those of other network communication paradigms. Below we describe the requirements ad hoc networks must meet.

Availability Availability ensures that the desired network services are available whenever they are needed. Systems that ensure availability seek to combat denial of service (DoS) and energy starvation attacks. In ad hoc networks, ensuring availability is perhaps more important than it is in traditional Internet. As all the devices in the network depend on each other to relay messages, DoS attacks are easy to perpetrate. For example, a malicious user could try to jam or otherwise try to interfere with the flow of information. Or else, the routing protocol should be able to malicious users by feeding the network with accurate information. There are routing protocols that can adjust well to the changing topology, but there are none that can defy all the possible attacks [Deng2002, Zhou 1999]. Another vulnerable point, which has no equivalence in traditional networks, is the limited battery power of wireless nodes. Normally, these nodes try to save energy with power saving schemes, so that when the device is not in active use, energy is not consumed. With battery exhaustion attacks, a malicious user can cause higher power consumption from other devices' battery, causing these devices to die prematurely [Stajanol999].

Authorization and key management Authorization is another difficult matter in ad hoc networks. As there is little or no infrastructure, identifying users (e.g., participants in a meeting room) is not an easy task. There are problems with rusted third party schemes and identity-based mechanisms for key agreement. A generic protocol for password authenticated key exchange is described in [Asokan2000]. It has several drawbacks even though it is possible to construct very good authentication mechanisms for ad hoc networks. A password uthenticated multi-party Diffie - Hellman key exchange seems to overcome many problems of the generic protocol.

Confidentiality and Integrity

Data confidentiality is a core security primitive for ad hoc networks. It ensures that the message cannot be understood by anyone other than the authorized personnel. With wireless communication, anyone can sniff the messages going through the air, and without proper encryption all the information is easily available. On the other hand, without proper authentication, it is difficult to enforce confidentiality. And if the proper authenticity has been established, securing the connection with appropriate keys does not pose a big problem. Data integrity denotes the immaculateness of data sent from one node to another. That is, it ensures that a message sent from node A to node B was not modified during transmission by a malicious node C. If a robust confidentiality mechanism is employed, ensuring data integrity may be as simple as adding one-way hashes to encrypted messages.

In addition to malicious 520 *AD* handle both the changing topology of the network and attacks from the attacks, integrity may be compromised because of radio interference, etc., so some kind of integrity protection is definitely needed for ad hoc networks.

Non-Repudiation Non-repudiation ensures that the origin of a message cannot deny having sent the message. It is useful for detection and isolation of compromised nodes. When a node A receives an erroneous message from a node B, non-repudiation allows A to accuse B using this message and to convince other nodes that B is compromised.

Security Solutions Constraints Historically, network security personnel have adopted a centralized, largely protective paradigm to satisfy aforementioned requirements. This is effective because the privileges of every node in the network are managed by dedicated machines - authentication servers, firewalls, etc. - and the professionals who maintain them. Membership in such a network allows individual nodes to operate in an open fashion - sharing sensitive files, allowing incoming network connections - because it is implicitly guaranteed that any malicious user from outside world will not be allowed access. Although these solutions have been considered very early in the evolution of ad hoc networks, attempts to adapt similar client-server solutions to a decentralized environment have largely been ineffective. To be efficiently applicable, security solutions for ad hoc networks should ideally have the following characteristics:

• **Lightweight:** Solutions should minimize the amount of computation and communication required to ensure the security services to accommodate the limited energy and computational resources of mobile, ad hoc-enabled devices;

• **Decentralized:** Like ad hoc networks themselves, attempts to secure them must be ad hoc: they must establish security without reference to centralized, persistent entities. Instead, security paradigms should levy the cooperation of all trustworthy nodes in the network;

• **Reactive:** Ad hoc networks are dynamic. Nodes - trustworthy and malicious - may enter and leave the network spontaneously and unannounced. Security paradigms must react to changes in network state; they must seek to detect compromises and vulnerabilities. Therefore, these solutions should be reactive;

• **Fault-Tolerant**: Wireless mediums are known to be unreliable; nodes are likely to leave or be compromised without warning. The communication requirements of security solutions should be designed with such faults in mind; they should not rely on message delivery or ordering.

Naturally, these are not stringent requirements: specific applications may relax some or all of the above based on their domain and the sensitivity of the information involved. Moreover, many ad hoc network applications do not require 2-party secure communication; instead, achieving broadcast or group security may be all that is needed.

KEY MANAGEMENT

Public key systems are generally recognized to have an upper hand in key distribution. In a public key infrastructure, each node has a public/private key pair. A node distributes its public key freely to the other nodes in the network; however it keeps its private key to only itself. A CA is used for key management and has its own public/private key pair. The CA's public key is known to every network node. The trusted CA is responsible to sign certificates, binding public keys to nodes, and has to stay online to verify the current bindings. The public key of a node should be revoked if this node is no longer trusted or leaves the network. A single key management service for an ad hoc network is probably not an acceptable solution, as it is likely to become Achilles' heel of the network. If the CA is unavailable, nodes cannot get the current public keys of other nodes to establish secure connections. In addition, if a CA is compromised, the attacker can sign erroneous certificates using the database of the private keys. Naive replication of CAs can make the network more vulnerable, since compromising of any single replica can cause the system to fail. Hence, it may be more prudent to distribute the trust to a set of nodes by letting these nodes share the key management responsibility.

SECURE ROUTING:

The contemporary routing protocols designed for ad hoc networks (discussed in Chapter 2) cope well with dynamically changing topology, but are not designed to provide defense against malicious attackers. In these networks, nodes exchange network topology in order to establish routes between them, and are another potential target for malicious attackers who intend to bring down the network. As for attackers, we can classify them into external and internal. External attackers may inject erroneous routing information, replay old routing data or distort routing information in order to partition or overload the network with retransmissions and inefficient routing. Compromised nodes inside the network are harder to detect and are far more detrimental. Routing information signed by each node may not work, as compromised nodes can generate valid signatures using their private keys. Isolating compromised nodes through routing information is also difficult due to the dynamic topology. Solutions must overcome these potential problems and use some properties of ad hoc networks to facilitate secure routing. Once the compromised nodes have been identified, if there is sufficient number of possibly disjoint and valid routes, the routing protocol should be able to bypass the compromised nodes by using alternate routes.

Cooperation in MANETs As opposed to networks using dedicated nodes to support basic networking functions such as packet forwarding and routing, in ad hoc networks these functions are carried out by all available network nodes. There is no reason, however, to assume that the nodes in the network will eventually cooperate with one another since network operation consumes energy, a particularly scarce resource in a battery powered environment like MANETs.

The new type of node misbehavior that is specific to ad hoc networks is caused by the lack of cooperation and goes under the name of node selfishness [Yoo2006]. A selfish node does not directly intend to damage other nodes with active attacks (mainly because performing active attacks can be very expensive in terms of energy consumption) but it simply does not cooperate to the network operation, saving battery life for its own communications. Damages provoked by selfish behavior can not be underestimated: a simulation study presented in [Michiardi2002a] shows the impact of a selfish behavior in terms of global network throughput and global communication delay when the DSR routing protocol is used. The simulation results show that even a small percentage of selfish nodes present in the network leads to severe performance degradation. Furthermore, any security mechanism that tries to enforce cooperation among the nodes ought to focus not only on one particular function, but on both the routing and the packet forwarding function. Schemes that enforce node cooperation in a MANET can be divided in two categories: the first is currency based [Yoo2005] and the second uses a local monitoring technique. The currency based systems are simple to implement but rely on a tamperproof hardware. The main drawback of this approach lies in establishing how the virtual currency has to be exchanged, making their use not realistic in a practical system. On the other hand, cooperative security schemes based on local monitoring of neighbors by each node, evaluating a metric that reflects nodes' behavior. Based on that metric, a selfish node can be gradually isolated from the network.

The main drawback of the second approach is related to the absence of a well-accepted mechanism that securely identifies the nodes of the network: any selfish node could elude the cooperation enforcement mechanism and get rid of its bad reputation just by changing its identity. The main research efforts addressing node selfishness problem are presented as follows. The CONFIDANT (Cooperation Of Nodes, Fairness In Dynamic Ad hoc NeT works) cooperation mechanism [Buchegger2002a, Buchegger2002b] detects malicious nodes by means of observation or reports about several types of attacks, thus allowing nodes to route around misbehaved nodes and thereby isolate them. CONFIDANT works as an extension to a routing protocol such as DSR. Here, nodes are provided with:

• A monitor for observations; Reputation records for first-hand and trusted second-hand observations about routing and forwarding behavior of other nodes;

• Trust records to control trust given to received warnings;
• A path manager to adapt their behavior according to reputation and to take action against malicious nodes. The dynamic behavior of CONFIDANT is as follows.

Nodes monitor their neighbors and change the reputation accordingly. If they have some reason to believe that a node is misbehaving, they can take action in terms of their own routing and forwarding and they can decide to inform other nodes by sending an ALARM message. When a node receives such an ALARM either directly or by promiscuously listening to the network, it evaluates how trustworthy the ALARM is based on the source of the ALARM and the accumulated ALARM messages about the node in question. It can then decide whether to take action against the misbehaved node in the form of excluding routes containing the misbehaved node, re-ranking paths in the path cache, reciprocating by non-cooperation, and forwarding an ALARM about the node. Simulations with nodes that do not participate in the forwarding function have shown that CONFIDANT can cope well, even if half of the network population acts maliciously.

Further simulations on the effect of second-hand information and slander have shown that slander can effectively be prevented while still retaining a significant detection speed-up over using merely first-hand information. The limitations of CONFIDANT lie in the assertion that the reputation is based on the detection. Events have to be observable and classifiable for detection, and reputation can only be meaningful if the identity of each node is persistent; otherwise it is vulnerable to spoofing attacks.

Token-Based

In an approach presented in [Yang2002], each node of the ad hoc network has a token in order to participate in the network operations, and its local neighbors collaboratively monitor to detect any misbehavior in routing or packet forwarding services. Upon expiration of the token, each node renews its token via its multiple neighbors: the period of validity of a node's token is dependent on how long it has stayed and behaved well in the network. A well-behaving node accumulates its credit and renews its token less frequently as time evolves.

The security solution is composed of four closely connected components:
• Neighbor verification: describes how to verify whether each node in the network is a legitimate or malicious node;
• Neighbor monitoring: describes how to monitor the behavior of each node in the network and detect occasional attacks from malicious nodes;
• Intrusion reaction: describes how to alert the network and isolate the attackers;
• Security enhanced routing protocol: explicitly incorporates the security information collected by other components into the ad hoc routing protocol.

In the token issuing/renewal phase, it is assumed a global secret key (SK)/public key (PK) pair, where PK is well known by every node of the network. SK is shared by k neighbors who collaboratively sign the token requested or renewed by local nodes. On the other hand, token verification follows three steps:

1) Identity match between the nodes's ID and the token ID,

2) Validity time verification,

3) Issuer signature. If the token verification phase fails, the corresponding node is rejected from the network and both routing and data packets are dropped for that node. Routing security relies on the redundancy of routing information rather than cryptographic techniques enforced by suitably modifying the AODV protocol and the Watchdog technique described earlier. However, the proposed solution possesses some drawbacks. The bootstrap phase needed to generate a valid collection of partial tokens, which are used by a node to create its final token, has some limitations.

For example, the number of neighbors necessary to complete the signature of every partial token has to be at least k, suggesting the use of such security mechanism in rather large and dense ad hoc networks. On the other side, the validity period of a token increases proportionally to the time during which the node behaves well, and this feature has less impact if node mobility is high. Frequent changes in the local subset of the network nodes who share a key for issuing valid tokens can cause high computational overhead, not to mention the high traffic generated by issuing/renewing a token, suggesting that the token-based mechanism is more suitable in ad hoc networks where node mobility is low. Spoofing attacks, where a node can request more than one token based on different identities, are not taken into account.

INTRUSION DETECTION SYSTEMS.

Each MH in an ad hoc network is an autonomous unit and is free to move independently. This implies that a node without adequate physical protection is susceptible to being captured or compromised. It is difficult to track down a single compromised node in a large network. Hence, every node in a wireless ad hoc network should be able to work in a mode wherein it trusts no peer. While intrusion prevention techniques such as encryption and authentication can reduce the risks of intrusion, they cannot be completely eliminated. Intrusion detection can be used as a second line of defense to protect network systems, because once an intrusion is detected, appropriate action can be put in place to minimize the damage, launch counter offensive measures, or even gather evidence for possible follow up prosecution.

Authentication Authentication denotes the accurate, absolute identification of users who wish to participate in the network. Historically, authentication has been accomplished by a well-known central authentication server. The role of the server is to maintain a database of entities, or users, and their corresponding unique IDs. The ID may be a digital certificate, public key, or both. Unfortunately the ad hoc paradigm does not accommodate a centralized entity creating protocol deployment issues.

Trusted Third Parties

One of the most rudimentary approaches to authentication in ad hoc networks uses a Trusted Third Party (TTP). Every node that wishes to participate in an ad hoc network obtains a certificate from a universally trusted third party. When two nodes wish to communicate, they first check to see if the other node has a valid certificate. Although popular, the TTP approach is laden with flaws. Foremost, it probably is not reasonable to require all ad hoc network-enabled devices to have a certificate. Secondly, each node needs to have a unique name. Although this is reasonable in a large internet, it is a bit too restrictive in an ad hoc setting. Recent research has introduced many appropriate variations of TTPs, and these are discussed later.

SENSOR NETWORK PLATFORMS AND TOOLS

A real-world sensor network application is likely to incorporate all the functionalities like sensing and estimation, networking, infrastructure services, sensor tasking, data storage and query. This makes sensor network application development quite different from traditional distributed system development or database programming. With ad hoc deployment and frequently changing network topology, a sensor network application can hardly assume an always-on infrastructure that provides reliable services such as optimal routing, global directories, or service discovery.

There are two types of programming for sensor networks, those carried out by end users and those performed by application developers. An end user may view a sensor network as a pool of data and interact with the network via queries. Just as with query languages for database systems like SQL, a good sensor network programming language should be expressive enough to encode application logic at a high level of abstraction, and at the same time be structured enough to allow efficient execution on the distributed platform. On the other hand, an application developer must provide end users a sensor network with the capabilities of data acquisition, processing, and storage. Unlike general distributed or database systems, collaborative signal and information processing (CSIP) software comprise reactive, concurrent, distributed programs running on ad hoc resource- constrained, unreliable computation and communication platforms. For example, signals are noisy, events can happen at the same time, communication and computation take time, communications may be unreliable, battery life is limited, and so on.

SENSOR NODE HARDWARE

Sensor node hardware can be grouped into three categories, each of which entails a different trade-offs in the design choices.

- **AUGMENTED GENERAL-PURPOSE COMPUTERS:** Examples include low-power PCs, embedded PCs (e.g. PC104), custom-designed PCs, (e.g. Sensoria WINS NG nodes), and various personal digital assistants (PDA). These nodes typically run –ff-the-shelf operating systems such as WinCE, Linux, or real-time operating systems and use standard wireless communication protocols such as IEEE 802.11, Bluetooth, Zigbee etc. Because of their relatively higher processing capability, they can accommodate wide variety of sensors, ranging from simple microphones to more sophisticated video cameras.
- **DEDICATED EMBEDDED SENSOR NODES:** Examples include the Berkeley mote family [1], the UCLA Medusa family [2], Ember nodes and MIT AMP [3]. These platforms typically use commercial off-the-shelf (COTS) chip sets with emphasis on small form factor, low power processing and communication, and simple sensor interfaces. Because of their COTS CPU, these platforms typically support at least one programming language, such as C. However, in order to keep the program footprint small to accommodate their small memory size, programmers of these platforms are given full access to hardware but rarely any operating system support. A classical example is the TinyOS platform and its companion programming language, nesC.
- **SYSTEM ON-CHIP (SOC) NODES:** Examples of SoC hardware include smart dust [4], the BWRC picoradio node [5], and the PASTA node [6]. Designers of these platforms try to push the hardware limits by fundamentally rethinking the hardware architecture trade-offs for a sensor node at the chip design level. The goal is to find new ways of integrating CMOS, MEMS, and RF technologies to build extremely low power and small footprint sensor nodes that still provide certain sensing, computation, and communication capabilities. Among these hardware platforms, the Berkeley motes, due to their small form factor, open source software development, and commercial availability, have gained wide popularity in the sensor network research.

SENSOR NETWORK PROGRAMMING CHALLENGES

Traditional programming technologies rely on operating systems to provide abstraction for processing, I/O, networking, and user interaction hardware. When applying such a model to programming networked embedded systems, such as sensor networks, the application programmers need to explicitly deal with message passing, event synchronization, interrupt handling, and sensor reading. As a result, an application is typically implemented as a finite state machine (FSM) that covers all extreme cases: unreliable communication channels, long delays, irregular arrival of messages, simultaneous events etc.

For resource-constrained embedded systems with real-time requirements, several mechanisms are used in embedded operating systems to reduce code size, improve response time, and reduce energy consumption. Microkernel technologies [7] modularize the operating system so that only the necessary parts are deployed with the application. Real-time scheduling [8] allocates resources to more urgent tasks so that they can be finished early. Event-driven execution allows the system to fall into low-power sleep mode when no interesting events need to be processed. At the extreme, embedded operating systems tend to expose more hardware controls to the programmers, who now have to directly face device drivers and scheduling algorithms, and optimize code at the assembly level.

Although these techniques may work well for small, stand-alone embedded systems, they do not scale up for the programming of sensor networks for two reasons:

- Sensor networks are large-scale distributed systems, where global properties are derivable from program execution in a massive number of distributed nodes. Distributed algorithms themselves are hard to implement, especially when infrastructure support is limited due to the ad hoc formation of the system and constrained power, memory, and bandwidth resources.
- As sensor nodes deeply embed into the physical world, a sensor network should be able to respond to multiple concurrent stimuli at the speed of changes of the physical phenomena of interest.

There no single universal design methodology for all applications. Depending on the specific tasks of a sensor network and the way the sensor nodes are organized, certain methodologies and platforms may be better choices than others. For example, if the network is used for monitoring a small set of phenomena and the sensor nodes are organized in a simple star topology, then a client-server software model would be sufficient. If the network is used for monitoring a large area from a single access point (i.e., the base station), and if user queries can be decoupled into aggregations of sensor readings from a subset of nodes, then a tree structure that is rooted at the base station is a better choice. However, if the phenomena to be monitored are moving targets, as in the target tracking, then neither the simple client-server model nor the tree organization is optimal. More sophisticated design and methodologies and platforms are required.

NODE-LEVEL SOFTWARE PLATFORMS

Most design methodologies for sensor network software are node-centric, where programmers think in terms of how a node should behave in the environment. A node- level platform can be node-centric operating system, which provides hardware and networking abstractions of a sensor node to programmers, or it can be a language platform, which provides a library of components to programmers.

A typical operating system abstracts the hardware platform by providing a set of services for applications, including file management, memory allocation, task scheduling, peripheral device drivers, and networking. For embedded systems, due to their highly specialized applications and limited resources, their operating systems make different trade- offs when providing these services.

For example, if there is no file management requirement, then a file system is obviously not needed. If there is no dynamic memory allocation, then memory management can be simplified. If prioritization among tasks is critical, then a more elaborate priority scheduling mechanism may be added.

UNIT – V

OPERATING SYSTEM: TINYOS

INTRODUCTION

Operating System: TinyOS

Tiny OS aims at supporting sensor network applications on resource-constrained hardware platforms, such as the Berkeley motes. To ensure that an application code has an extremely small foot-print, TinyOS chooses to have no file system, supports only static memory allocation, implements a simple task model, and provides minimal device and networking abstractions. Furthermore, it takes a language-based application development approach so that only the necessary parts of the operating system are compiled with the application.

To a certain extent, each TinyOS application is built into the operating system. Like many operating systems, TinyOS organizes components into layers. Intuitively, the lower a layer is, the „closer" it is to the hardware; the higher a layer is, the closer it is to the application. In addition to the layers, TinyOS has a unique component architecture and provides as a library a set of system software components. A components specification is independent of the components implementation. Although most components encapsulate software functionalities, some are just thin wrappers around hardware.

An application, typically developed in the nesC language, wires these components together with other application-specific components. A program executed in TinyOS has two contexts, tasks and events, which provide two sources of concurrency. Tasks are created (also called posted) by components to a task scheduler. The default implementation of the TinyOS scheduler maintains a task queue and invokes tasks according to the order in which they were posted. Thus tasks are deferred computation mechanisms. Tasks always run to completion without preempting or being preempted by other tasks. Thus tasks are non-preemptive.

The scheduler invokes a new task from the task queue only when the current task has completed. When no tasks are available in the task queue, the scheduler puts the CPU into the sleep mode to save energy. The ultimate sources of triggered execution are events from hardware: clock, digital inputs, or other kinds of interrupts. The execution of an interrupt handler is called an event context. The processing of events also runs to completion, but it preempts tasks and can be preempted by other events. Because there is no preemption mechanism among tasks and because events always preempt tasks, programmers are required to chop their code, especially the code in the event contexts, into small execution pieces, so that it will not block other tasks for too long.

Another trade-off between non-preemptive task execution and program reactiveness is the design of split-phase operations in TinyOS. Similar to the notion of asynchronous method calls in distributed computing, a split-phase operation separates the initiation of a method call from the return of the call. A call to split-phase operation returns immediately, without actually performing the body of the operation. The true execution of the operation is scheduled later; when the execution of the body finishes, the operation notifies the original caller through a separate method call.

In summary, many design decisions in TinyOS are made to ensure that it is extremely lightweight. Using a component architecture that contains all variables inside the components and disallowing dynamic memory allocation reduces the memory management overhead and makes the data memory usage statically analyzable. The simple concurrency model allows high concurrency with low thread maintenance overhead. However, the advantage of being lightweight is not without cost.

Many hardware idiosyncrasies and complexities of concurrency management are left for the application programmers to handle. Several tools have been developed to give programmers language-level support for improving programming productivity and code robustness.

IMPERATIVE LANGUAGE: NESC:

NesC [9] is an extension of C to support and reflect the design of TinyOS. It provides a set of language constructs and restrictions to implement TinyOS components and applications. A component in nesC has an interface specification and an implementation. To reflect the layered structure of TinyOS, interfaces of a nesC component are classified as provides or uses interfaces. A provides interface is a set of method calls exposed to the upper layers, while a uses interface is a set of method calls hiding the lower layer components. Methods in the interfaces can be grouped and named. Although they have the same method call semantics, nesC distinguishes the directions of the interface calls between layers as event calls and command calls. An event call is a method call from a lower layer component to a higher layer component, while a command is the opposite.

The separation of interface type definitions from how they are used in the components promotes the reusability of standard interfaces. A component can provide and use the same interface type, so that it can act as a filter interposed between a client and a service. A component may even use or provide the same interface multiple times.

COMPONENT IMPLEMENTATION

There are two types of components in nesC, depending on how they are implemented: modules and configurations. Modules are implemented by application code (written in a C-like syntax). Configurations are implemented by connecting interfaces of existing components. nesC also supports the creation of several instances of a component by declaring abstract components with optional parameters. Abstract components are created at compile time in configuration.

As TinyOS does not support dynamic memory allocation, all components are statically constructed at compile time. A complete application is always a configuration rather than a module. An application must contain the main module, which links the code to the scheduler at run time. The main has single StdControl interface, which is the ultimate source of initialization of all components.

DATAFLOW-STYLE LANGUAGE: TINYGALS:

Dataflow languages are intuitive for expressing computation on interrelated data units by specifying data dependencies among them. A dataflow diagram has a set of processing units called actors. Actors have ports to receive and produce data, and the directional connections among ports are FIFO queues that mediate the flow of data. Actors in dataflow languages intrinsically capture concurrency in a system, and the FIFO queues give a structured way of decoupling their executions.

The execution of an actor is triggered when there are enough input data at the input ports. Asynchronous event-driven execution can be viewed as a special case of dataflow models, where each actor is triggered by every incoming event. The globally asynchronous and locally synchronous (GALS) mechanism is a way of building event- triggered concurrent execution from thread-unsafe components. TinyGALS is such as language for TinyOS.

One of the key factors that affect component reusability in embedded software is the component composability, especially concurrent composability. In general, when developing a component, a programmer may not anticipate all possible scenarios in which the component may be used. Implementing all access to variables as atomic blocks, incurs too much overhead. At the other extreme, making all variable access unprotected is easy for coding but certainly introduces bugs in concurrent composition. TinyGALS addresses concurrency concerns at the system level, rather than at component level as in nesC. Reactions to concurrent events are managed by a dataflow-style FIFO queue communication.

TINYGALS PROGRAMMING MODEL

TinyGALS supports all TinyOS components, including its interfaces and module implementations. All method calls in a component interface are synchronous method calls- that is, the thread of control enters immediately into the callee component from the caller component.

An application in TinyGALS is built in two steps:

1. constructing asynchronous actors from synchronous components, and
2. constructing an application by connecting the asynchronous components through FIFO queues. An actor in TinyGALS has a set of input ports, a set of output ports, and a set of connected TinyOS components.
    a. An actor is constructed by connecting synchronous method calls among TinyOS components.
    b. At the application level, the asynchronous communication of actors is mediated using FIFO queues. Each connection can be parameterized by a queue size.
    c. In the current implementation of TinyGALS, events are discarded when the queue is full.

However, other mechanisms such as discarding the oldest event can be used.

NODE-LEVEL SIMULATORS:

Node-level design methodologies are usually associated with simulators that simulate the behavior of a sensor network on a per-node basis. Using simulation, designers can quickly study the performance (in terms of timing, power, bandwidth, and scalability) of potential algorithms without implementing them on actual hardware and dealing with the vagaries of actual physical phenomena. A node-level simulator typically has the following components:

SENSOR NODE MODEL: A node in a simulator acts as a software execution platform, a sensor host, as well as a communication terminal. In order for designers to focus on the application-level code, a node model typically provides or simulates a communication protocol stack, sensor behaviors (e.g., sensing noise), and operating system services. If the nodes are mobile, then the positions and motion properties of the nodes need to be modeled. If energy characteristics are part of the design considerations, then the power consumption of the nodes needs to be modeled.

COMMUNICATION MODEL: Depending on the details of modeling, communication may be captured at different layers. The most elaborate simulators model the communication media at the physical layer, simulating the RF propagation delay and collision of simultaneous transmissions. Alternately, the communication may be simulated at the MAC layer or network layer, using, for example, stochastic processes to represent low-level behaviors.

PHYSICAL ENVIRONMENT MODEL: A key element of the environment within a sensor network operates is the physical phenomenon of interest. The environment can also be simulated at various levels of details. For example, a moving object in the physical world may be abstracted into a point signal source. The motion of the point signal source may be modeled by differential equations or interpolated from a trajectory profile. If the sensor network is passive- that is, it does not impact the behavior of the environment-then the environment can be simulated separately or can even be stored in data files for sensor nodes to read in. If, in addition to sensing, the network also performs actions that influence the behavior of the environment, then a more tightly integrated simulation mechanism is required.

STATISTICS AND VISUALIZATION: The simulation results need to be collected for analysis. Since the goal of a simulation is typically to derive global properties from the execution of individual nodes, visualizing global behaviors is extremely important. An ideal visualization tool should allow users to easily observe on demand the spatial distribution and mobility of the nodes, the connectivity among nodes, link qualities, end- to-end communication routes and delays, phenomena and their spatio-temporal dynamics, sensor readings on each node, sensor nodes states, and node lifetime parameters (e.g., battery power). A sensor network simulator simulates the behavior of a subset of the sensor nodes with respect to time.

Depending on how the time is advanced in the simulation, there are two types of execution models: cycle-driven simulation and discrete-event simulation. A cycle-driven (CD) simulation discretizes the continuous notion of real time into (typically regularly spaced) ticks and simulates the system behavior at these ticks. At each tick, the physical phenomena are first simulated, and then all nodes are checked to see if they have anything to sense, process, or communicate. Sensing and computation are assumed to be finished before the next tick. Sending a packet is also assumed to be completed by then.

However, the packet will not be available for the destination node until next tick. This split-phase communication is a key mechanism to reduce cyclic dependencies that may occur in cycle-driven simulations. Most CD simulators do not allow interdependencies within a single tick.

Unlike cycle-driven simulators, a discrete-vent (DE) simulator assumes that the time is continuous and an event may occur at any time. As event is 2-tuple with a value and a time stamp indicating when the event is supposed to be handled. Components in a DE simulation react to input events and produce output events. In node-level simulators, a component can be a sensor node, and the events can be communication packets; or a component can be software module within and the events can be message passings among these nodes. Typically, components are causal, in the sense that if an output event is computed from an input event, then the time stamp of the output should not be earlier than that of the input event. Non-causal components require the simulators to be able to roll back in time, and worse, they may not define a deterministic behavior of a system.

A DE simulator typically requires a global event queue. All events passing between nodes or modules are put in the event queue and sorted according to their chronological order. At each iteration of the simulation, the simulator removes the first event (the one with earliest time stamp) from the queue and triggers the component that reacts to that event. In terms of timing behavior, a DE simulator is more accurate than a CD simulator, and as a consequence, DE simulators run slower. The overhead of ordering all events and computation, in addition to the values and time stamps of events, usually dominates the computation time. At an early stage of a design when only the asymptotic behaviors rather than timing properties are of concern, CD simulations usually require less complex components and give faster simulations. This is partly because of the approximate timing behaviors, which make simulation results less comparable from application to application, there is no general CD simulator that fits all sensor network simulation tasks. Many of the simulators are developed for particular applications and exploit application-specific assumptions to gain efficiency.

DE simulations are sometimes considered as good as actual implementations, because of their continuous notion of time and discrete notion of events. There are several open- source or commercial simulators available. One class of these simulators comprises extensions of classical network simulators, such as ns-2, J-Sim (previously known as JavaSim), and GloMoSim/ Qualnet.

The focus of these simulators is on network modeling, protocol stacks, and simulation performance. Another class of simulators, sometimes called software-in-the-loop simulators, incorporate the actual node software into the simulation. For this reason, they are typically attached to particular hardware platforms and are less portable. Example include TOSSIM [12] for Berkeley motes and Em* [13] for Linux-based nodes such as Sensoria WINS NG platforms.

THE NS-2 SIMULATOR AND ITS SENSOR NETWORK EXTENSIONS:

The simulator ns-2 is an open-source network simulator that was originally designed for wired, IP networks. Extensions have been made to simulate wireless/mobile networks (e.g. 802.11
MAC and TDMA MAC) and more recently sensor networks.

While the original ns-2 only supports logical addresses for each node, the wireless/mobile extension of it (e.g. [14]) introduces the notion of node locations and a simple wireless channel model. This is not a trivial extension, since once the nodes move, the simulator needs to check for each physical layer event whether the destination node is within the communication range. For a large network, this significantly slows down the simulation speed. There are two widely known efforts to extend ns-2 for simulating sensor networks:

SensorSim form UCLA and the NRL sensor network extension from the Navy Research Laboratory . SensorSim also supports hybrid simulation, where some real sensor nodes, running real applications, can be executed together with a simulation. The NRL sensor network extension provides a flexible way of modeling physical phenomena in a discrete event simulator. Physical phenomena are modeled as network nodes which communicate with real nodes through physical layers.

The main functionality of ns-2 is implemented in C++, while the dynamics of the simulation (e.g., time-dependent application characteristics) is controlled by Tcl scripts. Basic components in ns-2 are the layers in the protocol stack. They implement the handlers interface, indicating that they handle events. Events are communication packets that are passed between consecutive layers within one node, or between the same layers across nodes. The key advantage of ns-2 is its rich libraries of protocols for nearly all network layers and for many routing mechanisms. These protocols are modeled in fair detail, so that they closely resemble the actual protocol implementations. Examples include the following:

TCP: reno, tahoe, vegas, and SACK implementations. MAC: 802.3, 802.11, and TDMA. Ad hoc routing: Destination sequenced distance vector (DSDV) routing, dynamic source routing (DSR), ad hoc on-demand distance vector (AOPDV) routing, and temporarily ordered routing algorithm (TORA).
Sensor network routing: Directed diffusion, geographical routing (GEAR) and geographical adaptive fidelity (GAF) routing.

THE SIMULATOR TOSSIM:

TOSSIM is a dedicated simulator for TinyOS applications running on one or more Berkeley motes. The key design decisions on building TOSSIM were to make it scalable to a network of potentially thousands of nodes, and to be able to use the actual software code in the simulation. To achieve these goals, TOSSIM takes a cross-compilation approach that compiles the nesC source code into components in the simulation. The event-driven execution model of TinyOS greatly simplifies the design of TOSSIM. By replacing a few low-level components such as the A/D conversion (ADC), the system clock, and the radio front end,

TOSSIM translates hardware interrupts into discrete-event simulator events. The simulator event queue delivers the interrupts that drive the execution of a node. The upper-layer TinyOS code runs unchanged. TOSSIM uses a simple but powerful abstraction to model a wireless network. A network is a directed graph, where each vertex is a sensor node and each directed edge has a bit- error rate. Each node has a private piece of state representing what it hears on the radio channel. By setting connections among the vertices in the graph and a bit-error rate on each connection, wireless channel characteristics, such as imperfect channels, hidden terminal problems, and asymmetric links can be easily modeled.

Wireless transmissions are simulated at the bit level. If a bit error occurs, the simulator flips the bit. TOSSIM has a visualization package called TinyViz, which is a Java application that can connect to TOSSIM simulations. TinyViz also provides mechanisms to control a running simulation by, for example, modifying ADC readings, changing channel properties, and injecting packets. TinyViz is designed as a communication service that interacts with the TOSSIM event queue. The exact visual interface takes the form of plug-ins that can interpret TOSSIM events. Beside the default visual interfaces, users can add application- specific ones easily.

PROGRAMMING BEYOND INDIVIDUAL NODES: STATE-CENTRIC PROGRAMMING:

Many sensor network applications, such as target tracking, are not simply generic distributed programs over an ad hoc network of energy-constrained nodes. Deeply rooted in these applications is the notion of states of physical phenomena and models of their evolution over space and time. Some of these states may be represented on a small number of nodes and evolve over time, as in the target tracking problem, while others may be represented over a large and spatially distributed number of nodes, as in tracking a temperature contour. A distinctive property of physical states, such as location, shape, and motion of objects, is their continuity in space and time.

Their sensing and control is typically done through sequential state updates. System theories, the basis for most signal and information processing algorithms, provide abstractions for state updates, such as:

$$xk+1 = f(xk, uk)\ yk = g(xk, uk)$$

where x is the state of a system, u is the system input, y is the output and k is an integer update index over space and/or time, f is the state update function, and g is the output or observation function.

This formulation is broad enough to capture a wide variety of algorithms in sensor fusion, signal processing, and control (e.g., Kalman filtering, Bayesian estimation, system identification, feedback control laws, and finite-state automata).

However, in distributed real-time embedded systems such as sensor networks, the formulation is not as clean as represented in the above equations. The relationships among subsystems can be highly complex and dynamic over space and time. The following issues (which are not explicitly tackled in the above equations) must be properly addressed during the design to ensure the correctness and efficiency of the system. These issues, addressing where and when, rather than how, to perform sensing, computation, and communication, play a central role in the overall system performance. However, these „non-functional" aspects of computation, related to concurrency, responsiveness, networking, and resource management, are not well supported by traditional programming models and languages.

State-centric programming aims at providing design methodologies and frameworks that give meaningful abstractions for these issues, so that system designers can continue to write algorithms on top of an intuitive understanding of where and when the operations are performed. A collaborative group is such an abstraction. A collaborative group is a set of entities that contribute to a state update. These entities can be physical sensor nodes, or they can be more abstract system components such as virtual sensors or mobile agents hopping among sensors. These are all referred to as agents. Intuitively, a collaboration groups provides two abstractions: its scope to encapsulate network topologies and its structure to encapsulate communication protocols. The scope of a group defines the membership of the nodes with respect to the group.

A software agent that hops among the sensor nodes to track a target is a virtual node, while a real node is physical sensor. Limiting the scope of a group to a subset of the entire space of all agents improves scalability. Grouping nodes according to some physical attributes rather than node addresses is an important and distinguishing characteristic of sensor networks.

The structure of a group defines the “roles” each member plays in the group, and thus the flow of data. Are all members in the group equal peers? Is there a “leader” member in the group that consumes data? Do members in the group form a tree with parent and children relations?

For example, a group may have a leader node that collects certain sensor readings from all followers. By mapping the leader and the followers onto concrete sensor nodes, one can effectively define the flow of data from the hosts of followers to the host of the leader.

The notion of roles also shields programmers from addressing individual nodes either by name or address. Furthermore, having multiple members with the same role provides some degree of redundancy and improves robustness of the application in the presence of node and link failures.