ANMP: Ad hoc network Network Management Protocol*

Wenli Chen, Nitin Jain
Department of Computer Science
University of South Carolina
Columbia, SC 29208
chen,njain@cs.sc.edu

Suresh Singh†
Department of ECE
Oregon State University
Corvallis, OR 97331
singh@ece.orst.edu

Abstract

We present a protocol for managing mobile wireless ad hoc networks. The protocol uses hierarchical clustering of nodes to reduce the number of messages exchanged between the manager and the agents (mobiles). Clustering also enables the network to keep track of mobiles as they roam. ANMP is fully compatible with SNMPv3 and uses the same PDUs (Protocol Data Units) for data collection. The protocol also implements sophisticated security mechanisms that can be fine tuned to meet specific requirements. Finally, we have implemented the protocol along with a graphical user interface that allows a manager to change the view or specify management parameters on the fly.

1 Introduction

Ad Hoc networks are multi-hop wireless networks where nodes may be mobile. These types of networks are used in situations where temporary network connectivity is needed, such as in disaster relief or battlesite networks. In the disaster relief scenario, an ad hoc network would enable medics in the field to retrieve patient history from hospital databases (assuming that one or more of the nodes of the ad hoc network are connected to the Internet) or allow insurance companies to file claims from the field.

In this paper we focus our attention on the problem of *managing* ad hoc networks. By definition [17], *"network management is a process of controlling a complex data network so as to maximize its efficiency and productivity"*. This process involves data collection, data processing, data analysis and problem fixing. To accomplish this process, network management can be functionally divided into five areas defined by the International Organization for Standards(ISO) [17, 18]: fault management, configuration management, security management, performance management and accounting management. *Fault management* involves discovering, isolating and fixing problems in the network. This component of network management is responsible for ensuring smooth and continued operation of the network. *Configuration management* involves initialization and shutdown of the network. It also involves maintaining, adding and updating new network components. Part of the configuration module's function involves defining relationships between network entities. *Security management* involves controlling access to network components and information. This component is also responsible for implementing encryption and decryption schemes for secure end-to-end communication. *Performance management* involves collecting network statistics and tuning the network to improve performance. *Accounting management* involves tracking network utilization by various users and groups. This information can be very useful in network configuration and allocation of network resources to various groups in an organization.

Managing ad hoc networks is a significantly harder task then managing wireline networks because of several reasons. First, the topology of an ad hoc network may be quite dynamic, thus automated "on-the-move" reconfiguration is necessary. Second, the network is maintained by the combined efforts of all the mobile hosts themselves who often operate under severe constraints such as limited battery power, variable link quality, and limited storage capacity. As a result, we need to balance the management protocol performance against its overhead. Third, ad-hoc networks

*This work was supported by the NSF under grant number NCR-9706080 and ONR under grant number N00014-97-1-0806.
†Corresponding Author.

are employed for a diverse set of applications ranging from military battlesite networks to disaster relief applications. These different applications have different security requirements and the management protocol should be able to adapt easily to the characteristics of the situation. Finally, the management protocol should be compatible with current management protocols because there is an obvious need for interoperability.

In this paper we describe our management protocol ANMP that, we believe, provides a complete solution to the management problem. It is flexible and can be used in ad hoc networks where device characteristics are very diverse (e.g., devices may range from simple sensors to laptop and palmtop computers). The protocol also implements a flexible security mechanism based on SNMPv3 security and the security model used in the military for information classification. The specific ANMP features discussed in this paper are:

- Discussion of the network management goals for ad hoc networks, section 2.1;
- Security goals for ad hoc networks, section 2.1.3;
- Literature review, section 3;
- Design of ANMP, section 4;
- Architecture of ANMP, section 4.1;
- The security model, section 4.3;
- Clustering algorithms for management, section and 5;
- Description of management information collected for ad hoc networks, sections 7;
- The user interface, section 8.

The rest of the paper is organized as follows. In section 2 we describe, in more detail, the constraints imposed on network management in ad hoc networks. Section 3 provides an overview of existing network management protocols. Our conclusions are presented in section 9.

2 Challenges in Managing Ad Hoc Mobile Networks

Managing an ad hoc network well is frequently necessary because of the domain in which such networks are used. For instance, in a disaster scenario the management software ought to provide a complete picture of the deployment of relief crews, possibly overlaid on top of a map of the area containing information about potential hazards, victim density, etc. In this scenario, we may also imagine information from unmanned sensors (ground-based as in earthquake hit regions or airborne as in the case of nuclear disasters) being relayed to the management station providing a comprehensive picture of the situation. In this section, we first present the node and network properties of ad hoc networks in the context of network management. Then we discuss the different operational requirements of network management for these networks.

Some of the properties of ad hoc networks that make them difficult to manage well are (also see Table 1):

1. Nodes of an ad hoc network can range in complexity from simple sensors located in the field to fully functional computers such as laptops. An implication of this diversity is that not all nodes will be able to contribute equally to the management task. For instance, it is likely that sensors and small PDA (Personal Digital Assistant) type devices will contribute minimally to the task of management while more powerful machines will need to take on responsibilities such as collating data before forwarding it to the management station, tracking other mobiles in the neighborhood as they move, etc. Thus, the management protocol needs to function in very heterogeneous environments.

2. One mission of a network management protocol is to present the topology of the network to the network manager. In wireline networks, this is a very simple task because changes to the topology are very infrequent (e.g., a new node gets added, failure of a node or addition/deletion of a subnetwork, etc.). In mobile networks, on the other hand, the topology changes very frequently because the nodes move about constantly. Thus, the management station needs to collect connectivity information from nodes periodically. An implication of this is an increased message overhead in collecting topology information.

Property	*Implications for ANMP*
Wide variability in node capability	Inherently heterogeneous networks
Nodes are mobile	Need for topology updates
Battery operated	Minimize message/processing overhead
Possibility of partitions	Partitioned sub-networks need to operate autonomously
Variable link quality	Robust to high packet loss
Inherently insecure environment	Encryption is needed
Potential for node tampering	Build trust in an untrustworthy environment

Table 1: Properties of the ad hoc network environment.

3. Most nodes in ad hoc networks run on batteries. Thus, we need to ensure that network management overhead is kept to a minimum so that energy is conserved. Energy is consumed by a radio when a packet is transmitted or received (the DEC RoamAbout radio consumes approximately 5.76 watts during transmission, 2.88 watts during reception). In addition, the CPU expends energy in processing these packets. Thus, we need to reduce the number of packets transmitted/received/processed at each node. This requirement is contradictory to the need for topology update messages as discussed above.

4. Energy constraints and mobility can result in the network becoming partitioned frequently. For instance, nodes may power themselves off to conserve energy resulting in partitions or a node may move out of transmission range of other nodes. Similarly, a node may die when its battery runs out of power. In all these cases, the partitioned sub-networks need to continue running independently and the management protocol must be robust enough to adapt. For instance, when the network gets partitioned, the management protocol entities must quickly learn that the partition has occurred and reconfigure the sub-network(s) to function autonomously. Furthermore, when partitions merge, the management protocol must be capable of updating the network view without too much of an overhead.

5. Signal quality can vary quite dramatically in wireless environments. Thus, fading and jamming may result in a link going down periodically. An effect of this is that the network topology from a graph theoretic point-of-view changes. However, the physical layout of the network may not change at all. The management protocol must be able to distinguish this case from the case when node moves cause topology changes because, in the case of changing link quality/connectivity, it may not be necessary to exchange topology update messages at all. In order to be able to do this, the management protocol entity (which resides at the application layer) must be able to query the physical layer. This obviously violates the layering concept of OSI but it results in enormous savings (as we will see in section 5).

6. Ad hoc networks are frequently set up in hostile environments (e.g., battlesite networks) and are therefore subject to eavesdropping, destruction and possibly penetration (i.e., a node is captured and used to snoop). Thus the management protocol needs to incorporate encryption as well as sophisticated authentication procedures. We will discuss this more in the next section using examples.

2.1 Examples of some Operational Requirements for ANMP

In the introduction, we discussed the partitioning of network management functions into five categories. In this paper, however, we find it convenient to focus only on *data collection*, *configuration/fault management* and *security management* for ANMP. Before we discuss these three areas in detail, it is important to note that ad hoc networks, unlike wireline and cellular networks, are *temporary*. As a result the context of network management is quite different in these networks. For instance, a node failing because it has run out of battery power is treated as normal in ad hoc networks whereas a node failure in wireline networks is treated as a fault that must be corrected. In the remainder of this section we discuss operational requirements of ANMP (where it differs in its mission from existing network management protocols).

2.1.1 Data Collection

Data collection is a necessary function in ANMP where the management entities collect node and network information from the network. SNMP (Simple Network Management Protocol, see section 3.2) specifies a large list of information items that can be collected from each node. However, this list does not include some crucial data items that are specific to the ad hoc environment. We discuss these data elements in more detail in section 7 but some of the more interesting examples include status of the battery power (expected remaining lifetime), link quality, location (longitude and latitude), speed and direction, etc. All this information needs to be collected as and when it changes "significantly". For example, the location of a mobile node changes continuously but there is little point in updating it constantly because the overhead in message complexity is high. The best solution is to update this information when some other aspect of the node changes; for instance if the node's connectivity changes as a result of the motion then we may need to update its location, likewise if the node is expected to enter a hazardous location, we may require it to report its location frequently. We will discuss the methodology for implementing these types of *trap*-based updates in section 4.

One problem that arises in ad hoc networks in relation to data collection is the message overhead. Ad hoc networks have limited bandwidth (whose quality is variable) and we must ensure that the process of management does not consume significant amounts of this resource. Since network management runs at the application layer, the simplest way to implement data collection (at the manager station) is to poll each node individually. This method, unfortunately, results in a very high message overhead. This is easy to see if we model the ad hoc network as a graph. If the manager station is h_i hops away from a node i then the total message complexity (assuming equal sized messages) is $\sum_i h_i$ for each global update. A more efficient method of data collection is to use a spanning tree rooted in the manager station. We discuss this issue in detail in Sections 4.1 and 5.

2.1.2 Configuration/Fault Management

Unlike nodes in wireline networks, nodes in ad hoc networks either die, move or power themselves off to save energy. In all of these cases the network topology changes and the manager station needs to know the fate of these nodes. Table 2 lists some of the configuration changes that occur in ad hoc networks and possible actions the management protocol must take in these cases. In cases when a node is unavailable because of the reasons stated above, the manager records that fact in its database. However, even in the case when a manager knows that a node is dead, the entry for that node is not removed from the database. There are two reasons for this: first, the node may be resurrected if we put in a new battery (assuming that the node died because its battery ran out of power), and second, keeping in mind that ad hoc networks are temporary, we may need a complete history of the network's behavior to effect a redesign of protocols, evaluate security breaches, etc. In fact, a node may become unavailable many times during the life of the network and a complete log about the node's availability during this time needs to be maintained (not just an indication of its current status).

New nodes may join a network periodically (imagine a new set of volunteers who arrive at a disaster site to help) and these nodes must be incorporated seamlessly into the network. A network may also get partitioned periodically. In this case, we need to ensure that each partition selects its own manager. However when these partitions merge, one common manager needs to be chosen. Manager selection must be done based on the hardware and software capabilities of nodes and on the available battery power. One interesting scenario that can occur very easily is that of having geographically co-existing but independent networks. An example where this may happen is in the battlefield where a naval unit may be physically co-located with an infantry unit each of which has established its own ad hoc network. In this case, depending on security and other constraints, the two networks may decide to be managed together or continue being managed independently but exchange information such as link quality, presence of jamming, etc. with each other as an aid to better deploy the network resources. Finally, it is possible that a node may belong to two different networks and be managed by two (or more) managers. An example of this is can be drawn from the disaster relief model where a police officer may remain on a police (secure) ad hoc network but simultaneously be connected (and managed) to an ad hoc network of medical relief teams.

Any management protocol for ad hoc networks must be able to treat all of the above cases not as exceptions but as normal events. Further protocols that allow the manager to detect and react to the different configuration changes need to be developed. Today's management protocols do not have this level of flexibility and none of them can deal with most of the configuration changes listed above without external (user) intervention.

Configuration Changes	*Action at Network Manager*
Node dies	Mark node as dead
Node is disconnected	Mark node as disconnect, await reconnection event
Node is powered off	Mark node as unavailable
Node powers back on	Mark node as available
New nodes join the network	Add entry for these nodes
Network gets partitioned	New manager started in each partition
Partitions merge	One manager takes over (choice determined by hardware and software capability, remaining power
A new co-existing network detected	Managers acknowledge each other's presence see if the networks can be merged
Node multiply managed	Allow this condition in some cases

Table 2: Network Configuration changes.

	A trusts B *B trusts A*	*A trusts B* *B distrusts A*	*A distrusts B* *B trusts A*	*A distrusts B* *B distrusts A*
A and B are trustworthy	(SD,E)	(SD,N)	(UD,E)	(UD,N)
Only A is trustworthy	(SD,E) **Breach**	(SD,N) **Breach**	(UD,E)	(UD,N)
Only B is trustworthy	(UD,E)	(UD,N)	(UD,E)	(UD,N)
Neither is trustworthy	(UD,E)	(UD,N)	(UD,E)	(UD,N)

What is sent	SD – Secure digest	UD – Unclassified digest or nothing is sent
What is believed	E – Believe everything	N – Believe nothing

Table 3: Information transfer from $A \longrightarrow B$.

2.1.3 Security Management

Ad hoc networks are very vulnerable to security threats because the nodes (e.g., the unmanned nodes) can be easily be tampered with and signals can be intercepted, jammed or faked. Current protocols, such as SNMPv3 do provide us with some mechanisms to guard against eavesdropping, and replay attacks using secure unicast. However, there are some forms of communication for which we need to develop additional security.

In the previous section we discussed the possibility of co-located autonomous networks that may share some information with one another. The form of this information exchanged may vary depending on the level of trust these two networks have for one another and how trustworthy each of them is. Call these two subnetworks A and B and let us consider information transfer from A to B (the case for information transfer from B to A is identical). Table 3 indicates sixteen possible combinations than may occur. Each row represents one of four cases based on the trustworthiness of each subnetwork. Each column represents what is *believed* about one network by the manager of the other. Each entry in the table indicates the form of the information sent from A to B (secure or unclassified digest) and whether B believes the information or not. Finally, the table indicates two cases where there is a possibility of a security breach because A trusts B but B is not trustworthy. A management protocol must constantly monitor its own nodes to determine if they are trustworthy or not. This information needs to be used to determine if, overall, the data collected is trustworthy or worthless (e.g., an impersonator may deliberately misinform the manager). Likewise each manager must be able to determine if any of the networks it is exchanging data with are trustworthy or not.

Another security consideration in ad hoc networks concerns the process of data collection. Recall that in order to make data collection message efficient, it is desirable to collect information using a spanning tree. This allows an intermediate node to combine data from different nodes (one level down) before passing it on to a node on the next upper level. A problem with this model is the possibility of a security breach. In Figure 1, Sensor 1's data has a security clearance of 3 while node A has a clearance of 4 (smaller numbers represent higher security).

Thus, A should *not* be allowed to read data it forwards from Sensor 1. However, node B has a higher security clearance and can combine the data from Sensors 1&2 before forwarding it to the manager. Therefore, Sensor 1's data must be encrypted in a way that allows only B (and not A) to decrypt it. Today's management protocols do not look at information in this way (though some protocols do provide the capability to enforce end-to-end security on information).

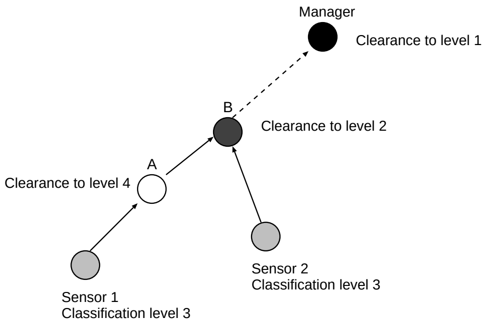


Figure 1: Informational security considerations.

2.2 Summary

From the discussion above it is clear that the management goals in ad hoc networks are quite different from that of traditional networks. This is because of the nature of applications utilizing ad hoc networks and because ad hoc networks tend to be temporary. ANMP has been designed to satisfy these new goals and in the following sections (after the literature review) we discuss how ANMP incorporates answers to the following questions:

- How to collect information efficiently?
- What type of information needs to be collected?
- How to handle dynamic network reconfiguration?
- How to delegate responsibility and collect information securely?
- How to build trust?

3 Literature Review

In this section we provide a survey of *architectures* and most commonly used *protocols* for network management. We first compare the merits of centralized, distributed, and hierarchical network management architectures. In section 3.3, we review some examples of network management protocols and examine SNMP in more detail in section 3.2.

3.1 Centralized, Distributed, and Hierarchical Network Management Architectures

Based upon the information collection and communication strategy, there are three types of network management architectures: centralized, distributed and hierarchical. In a centralized network management system, there is a single manager station which collects information from all nodes and controls the entire network. This is an easy implementation which, however, has some potential problems. For example, the manager station is a single point of failure. Also, if a network partition occurs because of some kind of failure, the portion which is disconnected from the manager is left without any management functionality. In an ad hoc network, a centralized architecture will suffer from a high message overhead in data collection. A distributed management system has multiple manager stations, each manages a subnetwork and communicates with other manager stations in a peer-to-peer manner. Using the distributed approach, a network management system could achieve higher reliability and efficiency as well as lower overhead both on communication and computation resources. Thus in a large telecommunication network, a distributed architecture is a better choice. This approach has been adopted by the Telecommunication Management

Network (TMN) and Management model for ATM networks [31]. Hierarchical network management systems use intermediate managers to distribute the manager tasks. Each intermediate manager has its domain, it collects and processes node information of its domain and passes the information to the upper level manager if necessary. It also distributes the messages from the upper level manager to nodes in its domain. There is no direct communication between intermediate managers. It is noteworthy that any of these architectures can be used in combination.

3.2 SNMP

In this section we give a brief description of Simple Network Management Protocol(SNMP). The description is important here because our management protocol, which is described later, uses many SNMP features. We chose SNMP because it is the most widely deployed management protocol standard on the Internet, it has a simple API (Application Programming Interface) and it runs over the TCP/IP protocol stack.

The bulk of network management functions involves gathering information from network elements. In SNMP, this information is represented in a structured manner in the Management Information Base(MIB). Every node in the network maintains a MIB which can have information about its current configuration, operation statistics and parameters to control its functioning. Objects in the MIB are divided into several groups[18]. Each group identifies a specific class of objects or variables that can be accessed by the management station. For example, the interface group contains statistics and configuration information about the physical interface of the entity. Objects in the MIB are defined using the BER (Basic Encoding Rules) associated with ASN.1. Each object can be one of the universal types[18]: INTEGER, OCTET STRING, NULL, OBJECT IDENTIFIER, SEQUENCE and SEQUENCE OF. The first four are primitive data types and form the building block for other types. The SEQUENCE and SEQUENCE OF types are used to define two-dimensional tables. These table structures are useful for storing information in an organized manner. Apart from the universal data types, there are few application data types[18] that can also be used in defining objects. For example: NetworkAddress, IpAddress, Counter, Gauge, TimeTicks and Opaque. There may be other data types, which can be defined in the standards as and when the need arises.

In a typical setup, a node is assigned the responsibility to manage a sub-network or a set of agents. This node, called the manager node, polls its agents by sending SNMP PDU's(Protocol Data Units)[18][20]. The management station can issue three types of PDU: *GetRequest, GetNextRequest* and *SetRequest.* The agent responds to these messages with *GetResponse* PDU only if it is successful in executing the request. For the Get requests, the agent sends back the requested value in the response PDU. For Set request, the agent sends back the new value of the target object in the set PDU. Agents can also unsolicitly send a *Trap* PDU to the management station informing about the occurrence of an exception condition. Since SNMP uses UDP as the underlying communication protocol, each PDU sent is treated as an independent request.

It is clear that the scheme described above will be efficient only if the management station is responsible for few agents. As this number grows, polling all the agents may eat up substantial amount of bandwidth and the manager's processing power. SNMP provides a scheme called *trap directed polling,* which could be used in such situations. In trap directed polling, the agents are configured in such a way that they inform the management station by sending trap messages only if some unusual event occurs. Upon receipt of such information, the management station can either choose to take some action or ignore it.

In a decentralized management architecture, SNMP provides some facilities for communication between managers. It defines another PDU called *InformRequest* for manager-to-manager communication. The receiving entity responds with a GetResponse PDU. SNMP has also defined a manager to manager(M2M)[18] MIB to configure SNMP entities acting in the manager's role. This MIB has been adopted from the RMON(Remote MONitor) alarm and event groups with some enhancements. The M2M-MIB has an alarm and event group. These groups are used to inform the management station or the destination station, about the occurrence of some critical event, thus shielding unnecessary details from other management stations and saving on network resources.

SNMPv3's security model comprises of two models:*User-based Security Model(USM)* [22] and *View-based Access Control Model(VACM)* [23]. USM deals with security in the transmission/receipt of messages between two end-points. It provides for authentication, encryption and timeliness check. The VACM module dictates the access control on SNMP MIB objects by various entities. To support access control, SNMPv3 defines *contexts, views* and *groups.* A Context is a collection of management information accessible by a SNMP entity(stored in the context table). Views are collection of MIB sub-trees(stored in view table). A view can be of read, write or notify type. A Group is defined as a set of zero or more users represented by their security name, who share the same access rights (stored in the group table).

3.3 Examples of Other Network Management Systems

CMIP [17, 18]: Common Management Information Protocol was proposed as a standard, to superseed SNMP. The standards are exhaustive in their specifications and include almost every possible detail for network management function. CMIP was specified to work over the OSI [29] protocol stack. The drawbacks of CMIP are: complex to use and implement, requires a lot of resources to execute, high overhead and the fact that few networks use the OSI protocol suites.

LMMP [17]: LAN Man Management Protocol was developed as a network management solution for LANs. LMMP was built over the IEEE 802 Logical Link Layer(LLC). Therefore it is independent of network layer protocol. Functionally it is equivalent to Common Management Information Service over IEEE 802 LLC (CMOL). The advantages: easier to implement and protocol independent. The disadvantages: LMMP messages cannot go beyond a LAN boundary because it does not use any network layer facilities. LMMP agents are also not dynamically extensible.

TMN [30]: Telecommunications Management Network was built over the OSI reference model, but it includes support for Signaling System No. 7, TCP/IP, ISDN, X.25 and 802.3 LAN - based networks. TMN has some advantages over the existing standards. For instance, TMN supports a larger number of protocols suites, incorporates features from existing management protocols, wider domain of acceptance and is "future-proof". The disadvantages are: large amounts of processing required and agents are not extensible at run time.

4 Overview of ANMP

In designing ANMP, we concentrated on developing a message efficient, secure protocol that addresses the issues raised in section 2. Our focus has been on developing a lightweight protocol that is compatible with SNMP. We believe that this is necessary because: 1) SNMP is a widely used management protocol today, and, 2) ad hoc networks can be viewed as extensions of today's networks used to cover areas lacking a fixed infrastructure. In operation, it is quite likely that an ad hoc network would be connected to a local area wireline network (using a gateway). In such cases, we can configure the ANMP manager to be viewed either as a peer of the SNMP manager (which is managing the wireline network) or as an agent of the SNMP manager. This flexibility, we believe, is a major strength of ANMP. Obviously ANMP can operate in isolated ad hoc networks as well. In this section we provide an overview of the major design choices made in ANMP. However, we would like to note a couple of implementation details first:

- The PDU (Protocol Data Unit) structure we use is identical to SNMP's PDU structure
- UDP is the transport protocol used for transmitting ANMP messages
- Lost data is not retransmitted by ANMP because information is periodically updated anyway. Furthermore, if the application sitting on top of ANMP wishes to obtain the lost information, it can request the ANMP manager to solicit that information again.

4.1 ANMP Architecture

In order to have a protocol that is message efficient, a hierarchical model for data collection is appropriate since intermediate levels of the hierarchy can collate data (possibly producing a digest) before forwarding it to upper layers of the hierarchy. A problem, however, with utilizing a hierarchical approach in ad hoc networks is the cost of *maintaining* a hierarchy in the face of node mobility. A good tradeoff we observed is to use a 3-level hierarchical architecture for ANMP. Figure 2 illustrates this architecture. The lowest level of this architecture consists of individual managed nodes called *agents*. Several agents (that are close to one another) are grouped into clusters and are managed by a *cluster head* (the nodes with squares around them in the figure). The cluster heads in turn are managed by the network manager. It is important to emphasize two points here:

- Clustering for management (as in ANMP) is very different from clustering for routing, as we discuss in section 5.
- A manager is frequently more than one hop away from the cluster heads (Figure 2 is a logical view and not a physical view).

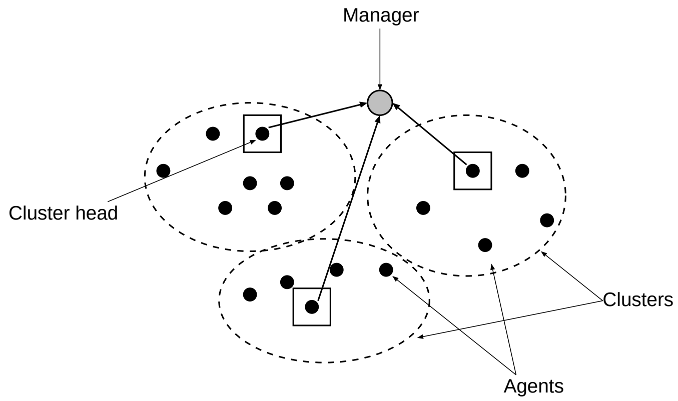


Figure 2: ANMP's hierarchical architecture.

The structure of the clusters is dynamic. Thus, as nodes move about, the number and composition of the clusters changes. Similarly, the nodes serving as cluster heads also changes over time. In section 5 we present two different algorithms for forming and maintaining clusters. The first algorithm views the ad hoc network as a graph (where two nodes within transmission range of one another have an edge) and forms clusters with the property that all nodes within a cluster are one or two hops away from the cluster head. The second algorithm uses latitude and longitude information to form clusters based on the spatial density of nodes. Both of these algorithms have been optimized such that the clusters formed have the following properties:

- The clusters are neither too large nor too small. Large clusters defeat the purpose of using a hierarchical structure to collect data since the message overhead of collecting data within large clusters will be high. Likewise, if we have very small clusters then there will be many cluster heads all of which will be controlled by the manager. Thus, the message overhead in transactions between the cluster heads and the manager is high.
- The clusters are formed such that node mobility does not result in frequent recomputation of the clusters. This property is necessary if we are to reduce the message overhead of *maintaining* clusters.
- As we will see, it is sometimes the case that nodes move out of one cluster and into another but are not incorporated into the new cluster immediately. This is because cluster maintenance algorithms only run periodically and not continuously (to keep message costs low). The effect of this is that some percentage of nodes may be *unmanaged by cluster heads* for short periods of time. This is not really a problem (except for the message overhead in data collection) because *these nodes are still in communication with the overall manager and can be directly managed by the manager.*

The selection of cluster heads in our protocols is based on node ID and node placement within the cluster. However, other methods of selecting cluster heads, such as remaining battery power, can also be used to select cluster heads. Our algorithms place no restriction on this choice. In section 5 we study the performance of the clustering algorithms. Specifically, we examine the message overhead of maintaining clusters as a function of node speed and the tradeoff between the number of nodes unmanaged by clusterheads as a function of message overhead.

4.2 Data Collection and Control

ANMP extends the MIBs used in SNMP to include ad hoc network specific information. As we noted in section 2, the information needed in ad hoc networks is usually very different from the information needed in wireline networks. We describe the MIB structure in detail in section 7. In this section, we outline the method of data collection and the control portion of ANMP that the manager can use to influence the behavior of the agents.

Every node populates the fields of the MIB structures based on what the manager wants. For example, a node puts in its latitude and longitude, remaining battery power information etc. into MIBs and sends this information to the cluster head. The cluster head collects information from all its agents (nodes), then prepares a summary or simply concatenates the information for transmittal to the overall manager. To maintain all this information, the cluster head keeps *tables* of MIBs with entries for all agents. This method is identical to the manner in which SNMP agents transmit information to their manager and the way in which managers exchange information. As we noted earlier, we use SNMP PDUs to exchange information between ANMP entities.

One novel feature of ANMP is the ability of the manager to exercise full control on the behavior of the agents. Let us look at a simple example. Since most nodes in an ad hoc network operate on battery power, their operation should be optimized to maximize the lifetime of the battery. This can be done by putting the node in a power-saving state (sleep mode) when there is no user activity. Now, assume that a node of the ad hoc network is in a region where there is a high probability of danger to the user. In such a circumstance, it is unwise for the user's machine to enter sleep mode (even if the user is not using it) because information about the danger (if it occurs) needs to be sent to the manager. In this case, ANMP has a facility by which the manager can force the device to remain active and not power down. In its general form, we implement this functionality in ANMP as follows. As in SNMP, we use *alarms* that are triggered when the value of a variable crosses some threshold. This typically causes the agent to report some statistic to the manager (e.g., too many error packets in a row). In ANMP, we associate a *function* with each alarm. The function can be as general as we want it to be and can alter node behavior. Furthermore, an agent or a manager can dynamically download a different function to associate with an alarm. In the example above, the alarm is triggered when the node detects no user activity for x minutes. The typical behavior is for the node to enter sleep. However, the manager can download a function (and bind it with the alarm) that forces the node to remain awake or operate the video camera (say) or do any one of a wide variety of activities. In summary, then, we see that *a manager in ANMP can completely reconfigure the nodes of the ad hoc network.*

4.3 Security in ANMP

ANMP implements the unicast security of SNMPv3. In addition however, ANMP supports *secure multicasts* and the military security model. Consider an application of secure multicasts in the ad hoc environment. Say the manager needs to configure a set of sensors to track a chemical cloud. To ensure that the message is not intercepted by intruders, it needs to be transmitted securely (i.e., encrypted, with timeliness information to prevent replay attacks and with a digital signature). In SNMPv3, this can be done by transmitting individual messages from the manager to each of the sensors. However, a secure multicast is a better alternative since it reduces the message overhead. We will discuss ANMP's implementation of secure multicasts in section 6.1.

ANMP also implements the military security model where we assign security clearances to nodes. Likewise, we assign security *classification levels* to data. Cluster heads are unable to read MIB data that is at a higher classification level. They can however read data at classification levels equal to their own or lower. A problem that we need to solve, in order to enforce this mechanism, is ensuring that an agent knows the security clearance of its cluster head that it uses to determine how to encrypt its data (see Figure 1). In ANMP, the manager assigns security levels to nodes and this information is distributed as an unforgeable ticket to cluster heads. Cluster heads transmit this information to all their cluster nodes when the clusters are formed. We discuss this security feature and the multicast security model in more detail in section 6.

5 Clustering Algorithms

Forming clusters is the most logical way of dividing an ad hoc network to simplify the task of management. Such a division facilitates decentralized network management that is more fault tolerant and message efficient. To keep the message overhead low our protocol, we don't try to keep track of every movement of all the nodes in the network. We believe that the overhead in collecting such information is not justified and in most cases not required for management applications. Rather, we respond to movements only when they result in significant topology change. We have developed two different clustering approaches: graph-based clustering and geographical-based clustering. The first approach models the ad hoc network as a graph and forms clusters based on the graph topology. The second approach uses GPS (Global Positioning System) information to divide the network into clusters. In both algorithms, we strive to ensure a low message overhead and form clusters in such a way that it does not induce frequent cluster changes.

Before we describe our two clustering algorithms in detail, we believe that it is important to restate our reason for using clusters and to draw a distinction between our use of *clusters for management* vs *clusters for routing*. We use clustering in ANMP to logically divide the network into a 3-level hierarchy in order to reduce the message overhead of network management. Since ANMP is an application layer protocol, ANMP presupposes the existence of an underlying routing protocol. Thus, the manager node can always reach any of the nodes in the ad hoc network (so long as they both lie in the same partition) and can manage them directly. Clustering simply introduces intermediate nodes called clusterheads for the purpose of reducing the message overhead. Thus, our clustering algorithms serve a *weaker and different objective* when compared with clustering algorithms used for routing.

In cluster-based routing (see for example [25, 33, 36] etc.), neighboring nodes form clusters and select a clusterhead. Clusterheads have the responsibility of routing packets sent to nodes outside the cluster. It is easy to see that clustering here serves a very fundamental goal of maintaining routes. Finally, we note that *if* the underlying routing protocol is cluster-based, ANMP could simply use these clusters for management as well. However, if the routing protocol is not cluster-based (see [34, 35]), the two clustering algorithms we describe form clusters and rely on routing support to exchange control messages.

5.1 Graph-based Clustering

In this section we first describe the basic graph-based clustering algorithm. The following section discusses the *maintenance* algorithm that deals with node mobility after clusters have formed (cluster changes caused due to local node mobility). We discuss the performance of the algorithm in section 5.1.2.

A node in the graph represents a mobile host in the network. There is an undirected link between two nodes if they are within transmission range of each other. For the purpose of clustering we assume the following:

- each node in the network has a unique ID,
- each node in the network maintains a list of its one hop neighbors (recall that ANMP runs at the application layer and therefore can obtain this information from the network or MAC layer),
- messages sent by a node are received correctly by all its neighbors within a finite amount of time (this requirement is not essential to the operation of the algorithm, however).

In our algorithm the node with *minimum ID* among its neighbors, which have not joined any other cluster, forms a cluster and becomes the *cluster head*[1]. Each node that has not yet joined any cluster, upon hearing from a cluster head, declares that it has joined a cluster. If any node has been included in more than one cluster then all but one cluster head prunes the node entry from their list when they do not hear any information from that node.

Figure 3 illustrates cluster formation in a simple ad hoc network. The node with the minimum ID forms the cluster and becomes the cluster head. Here node 1 is the minimum ID node among its neighbors, therefore, it forms a cluster C1. Node 4 does not initiate cluster formation because it is not the minimum ID node among its neighbors. When node 2 broadcasts a message that it has joined cluster C1, node 4 realizes that it is now the minimum ID node. Since the cluster formation runs distributedly, it is possible that by the time node 4 receives the broadcast message from node 2, node 3 has already initiated cluster formation and node 5 also sends a message that it has joined some cluster C3. Thus when node 4 starts cluster formation, it only includes the remaining nodes among its neighbors and froms C2.

It may be noted that because the node with minimum ID considers only its one hop neighbors while forming the cluster, the nodes in a cluster are one hop away from the cluster head and at most two hops away from any other cluster mate when the cluster is formed. The information maintained by each node after clusters are formed is:

- *Neighbor list*: list of nodes that are one hop away,
- *Cluster list*: list of all nodes that are its cluster mates,
- *Ping Counter*: indicates the time steps elapsed since it last heard from its cluster head.

5.1.1 Cluster Maintenance for Graph-based Clustering

Since the nodes of an ad hoc network may be mobile, the topology of the network changes over time. As a result the clusters and cluster membership has to be updated. One noteworthy feature, however, is that all cluster membership changes will be local to clusters in some geographical region. This means that we do not need to recompute cluster membership for the entire network.

[1]Different ways of assigning these IDs result in different optimizations. For instance, say that a node's life is represented by a k bit number. Then an ID of the form EH where E is the current age of the node (a k bit number) and H its hardware address results in cluster heads who are relatively young.

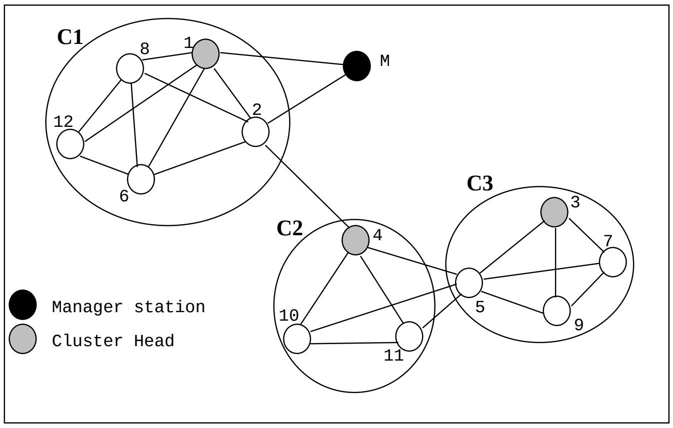
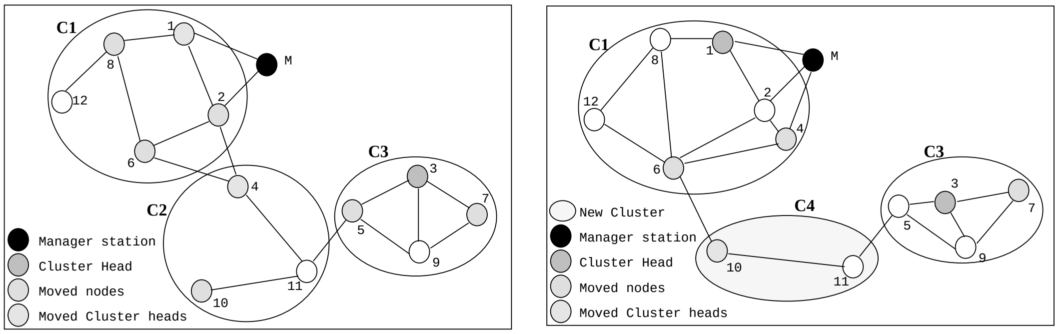


Figure 3: Clusters formed using graphical clustering.

Changes in cluster membership are triggered when a node moves out of a cluster (and into another) or when the cluster head itself moves out of a cluster (or, relatively speaking, cluster members move away from the cluster head). Figure 4 presents some illustrative cases. Figure 4(a), shows a situation where nodes move about but are still connected to at least one of their cluster mates (see Figure 3). Here there is no need to recompute clusters. Figure 4(b) shows two scenarios. First, when a node moves across the cluster boundary and second when the cluster head gets disconnected. It can be seen that node 4 gets disconnected from all the members of its previous cluster. Since node 4 is two hops away from the cluster head of cluster C1, it sends a join request to node 2. On receiving such event from node 4, node 2 adds node 4 to its cluster list and broadcasts it to all the members. Meanwhile, nodes 10 and 11 discover that their cluster head has moved away, they initiate cluster formation and form a new cluster C4.



(a) (b)

Figure 4: Effect of node mobility on clusters.

The example above indicates an important property of our maintenance algorithm:

```
When new clusters are formed, all nodes in the cluster are one hop
away from the cluster head. However, as nodes move about, we allow
nodes to join clusters even if they are two hops away from the cluster
head of an existing cluster. This flexibility drastically reduces the
message overhead of the algorithm.
```

Let us now examine our maintenance algorithm in some more detail. The purpose of this is to maintain clusters while minimizing the number of messages generated due to topology change. If a node is connected to at least one node in its cluster list, it is still considered to be part of that cluster. If a node detects that it has no links to any of its previous cluster mates, it either forms a new cluster by itself or joins another cluster. The crux of our algorithm is to differentiate between node movements within the cluster and movements across the cluster boundary. We identify four types of events that a node can detect as a result of mobility. A node can detect:

- a new neighbor, who is also a cluster mate,
- a previous neighbor and cluster mate has moved,
- that it was previously directly connected to the cluster head but is no longer directly connected or it was previously not directly connected but is now directly connected,
- a new neighbor, who wants to join the cluster.

At every fixed interval, called time steps, each node locally creates a list of events that it has observed and sends it to its cluster head. The cluster heads collects these events and re-computes the cluster list. If there is any change in the cluster membership, the cluster head broadcasts the new list. Thus, whenever a node moves out of a cluster or joins a cluster the message exchange is restricted to within that cluster. In order to minimize the number of cluster changes, we allow a node to join a cluster if the cluster head is two hops away. The restriction of two hops (as opposed to, say, three hops) has been enforced to avoid the creation of big clusters.

In such a division of the network, the cluster head plays an important role. That is why a major event that can occur is the *movement of the cluster head.* To determine if the cluster head has moved away, we use a *ping* counter at each node. This counter gets incremented at every time step. If the counter at the cluster head crosses some threshold value then the cluster head sends a ping message to all its cluster mates indicating that it is still alive. If the cluster mates do not hear a ping after their ping counters cross a threshold, they assume that the cluster head is either dead or has moved out. Once a node detects such an event, it cannot be certain about its cluster list. New cluster(s) are formed with one-hop neighbors in the same way as we initially form the clusters. It is easy to see that the frequency at which these pings are sent plays an important role in maintaining the clusters. If the ping frequency is small, then between consecutive pings, some nodes may be *unmanaged by a clusterhead* (i.e., they do not belong to any cluster). Unfortunately, even though a high frequency of pings minimizes the number of nodes unmanaged by clusterheads, it results in a higher message overhead. We examine this tradeoff between the percentage of nodes unmanaged by clusterheads and ping intervals in section 5.1.2.

One method of reducing the number of ping messages while simultaneously keeping the fraction of nodes unmanaged by clusterheads small is to exploit information available at the MAC layer. The MAC layer in wireless networks periodically transmits a beacon to announce itself to its neighbors. Thus the MAC layer keeps an updated list of its one-hop neighbors. If nodes transmit this list to their neighbors, changes in the cluster membership can be detected quickly. If the cluster head moves away, its departure will be noticed by the MAC layer of its one-hop neighbors. These nodes can act on this information to quickly re-form clusters. We will show in section 5.1.2 that this simple optimization does indeed improve performance.

Another characteristic of ad hoc networks are *partitions.* If a sub-network gets partitioned from the main network, it is treated as any other cluster because our protocol does not require any information exchange between clusters. If a single node gets partitioned it forms a cluster by itself. However, when it gets reconnected it tries to join another cluster. We believe that clusters of too small or too large a size are both inefficient from the point of view of network management.

5.1.2 Performance of Graph-based Clustering

We conducted experiments for 30, 60, 90 and 120 node ad hoc networks though we present the results for the 30-node case only (the results for the other cases is similar). To study the performance of this clustering algorithm, we simulate an ad hoc network in which the 30 nodes move randomly within a 1500x1500 unit box (for 60 nodes, the area of the playground was 2x and so on). Each node selects a direction randomly uniformly in the range $[0, 360]$ degrees. It moves along that direction for a length of time that is exponentially distributed. At the end of this time, it recomputes the direction and travelling time. When a node hits the edge of the box, it wraps around and emerges from the opposite edge. In our simulation we use the same transmission power for all the nodes in the network and represent the ad hoc network as an undirected graph with links between two nodes if they are within transmission range of each other. The average speed of nodes ranges from 1 unit/sec to 50 units/sec in different runs. The transmission range of a node is fixed at 450 units. Each reading is an average of ten readings and the simulation time is 1000secs. Finally, we assumed a packet loss probability of 10^{-3} in all simulations.

Figure 5 shows the message overhead of our protocol for the case when we use only pings to detect topology changes (graph on the left) and when we use pings along with MAC layer information (the graph on the right). The x-axis indicates the speed in units per second and the y-axis shows the number of messages exchanged per second for cluster maintenance. Each curve in the graph depicts the message overhead incurred for different ping intervals, which vary from no ping (equivalent to infinite time steps) down to 1 ping message every time step. It may be noted that message overhead increases almost linearly with increase in speed. The message overhead when we use MAC layer information shows similar behavior, but there are more messages exchanged. This is obvious because of the fact that nodes do not wait for the ping message from the cluster head. The clustering algorithm is triggered as soon as the nodes detect cluster head movement.

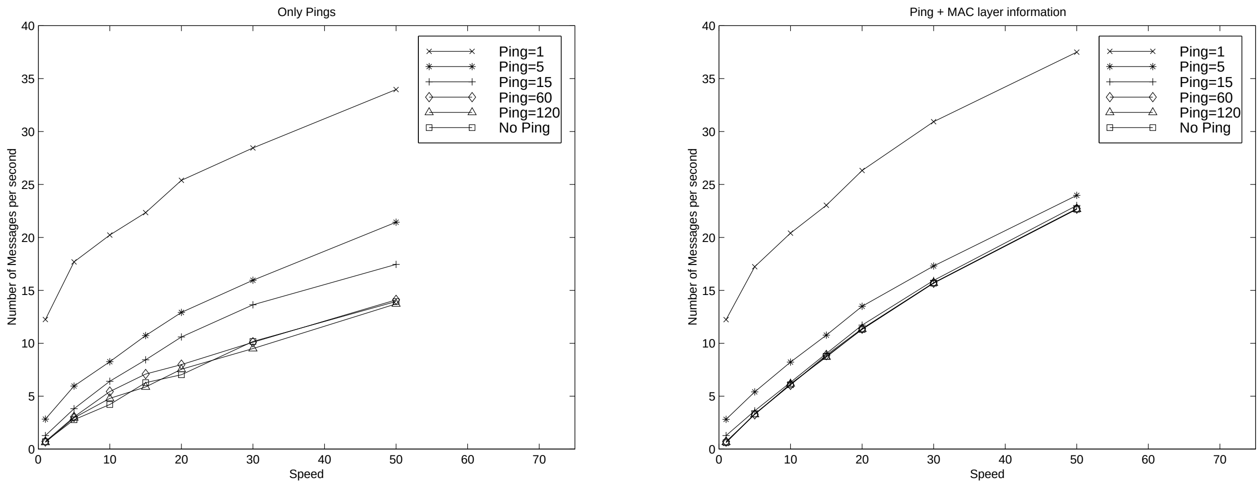


Figure 5: Message Overhead for Graphical Clustering.

Figure 6 shows the percentage of nodes unmanaged by clusterheads. The plot on the left is for the case when we only use pings to detect topology changes while the graph on the right is for the case when we use pings as well as MAC layer information to detect topology changes. It can be clearly seen from the graph that at higher speeds the percentage of nodes unmanaged by clusterheads goes up. This is mainly because of frequent disconnections and partitions generated in the network. Interestingly, the percentage of nodes unmanaged by clusterheads is as high as 50-70% when we only use pings but stays below 10% when we use MAC layer connectivity information as well. This improvement comes about because cluster head movement is detected earlier in the second case. At higher speeds the probability of cluster head leaving the cluster is high. In such a case the nodes in that cluster are considered to be unmanaged by clusterheads until the next ping message becomes overdue (if we do not use MAC layer information).

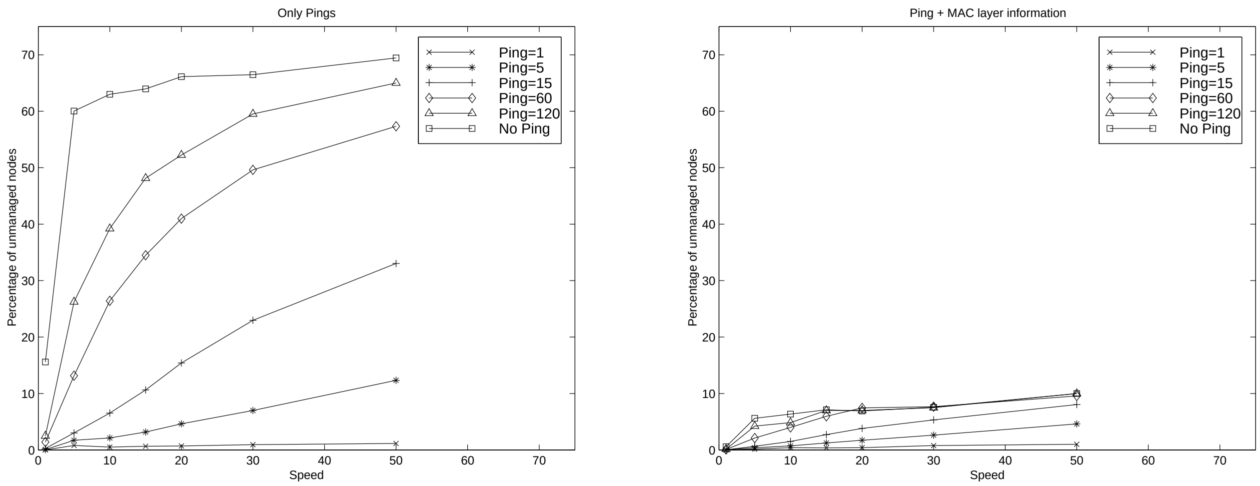


Figure 6: Percentage of nodes unmanaged by clusterheads.

From the results it can be seen that with the modified version of the protocol we can manage almost 90% of nodes in the network without using the ping message. However, if we choose to use the unmodified version of the protocol we can manage 90% of the network by using a ping interval of 5 units. The message overhead in this case, interestingly, is almost identical to the message overhead for the modified protocol (with MAC information).

5.2 Geographical Clustering

This protocol takes advantage of the Global Positioning System (GPS). GPS is a satellite-based radio-navigation system developed and operated by the U.S. Department of Defense. GPS lets the land, sea, and airborne users know their three dimensional position, velocity, and time, 24 hours a day in all weather conditions, anywhere in the world. The GPS system can achieve an accuracy of the order of meters (see [16] for more detail about GPS).

In this algorithm we split the nodes into clusters based on their spatial density. Each cluster is a rectangular box and one of the more centrally located nodes within that box is selected as the cluster head. Node mobility tends to change the spatial density of the nodes and therefore, over time, the clusters will need to be reformed. Our clustering algorithm works in two stages to deal with problems caused by node mobility:

- A periodic procedure forms clusters for the entire network (this is done by the manager) using spatial density as a guide,
- During the time between two executions of the periodic protocol, we run another algorithm to deal with node mobility. This algorithm is executed locally within each cluster. Some outcomes of this procedure include changes in cluster membership (add new members or remove departed members), changes in cluster head responsibility (when a cluster head moves away), merging two clusters into one (if the spatial density in each is small) and splitting dense clusters into smaller ones.

The remainder of this section is organized as follows. In section 5.2.1 we outline the periodic algorithm that runs globally, section 5.2.2 discusses the maintenance algorithm that runs between two consecutive executions of the periodic algorithm. Section 5.2.3 presents the performance of this clustering algorithm.

5.2.1 Central Periodic Clustering Algorithm

The manager of the network collects location information from all nodes periodically (this is not an overhead because the manager has this location information for all nodes as part of its management task). This information is used by the manager to create clusters. Before we discuss the algorithm itself, however, let us look at an example of how the algorithm works. In Figure 7 we consider a small ad hoc network consisting of twenty nodes. In the first step, the ad hoc network box is divided into vertical and horizontal strips. The number of nodes that lie in each strip is counted and we construct a bar graph (as shown in the figure) for the horizontal and for the vertical dimension. Next, we look for *valleys* in the bar graph (these represent sparse areas) and split the box along those valleys. This is done along both the horizontal and the vertical direction. As a result, nodes in areas where there is a high density of nodes fall in the same cluster. A problem with the above algorithm is that some clusters may be very large or very small. Therefore, after we form the clusters using the procedure above, we either split large clusters or merge smaller ones into larger ones.

Let us now look at the algorithm in more detail:

1. *Box generation*
   (a) Compute the distribution of mobile hosts along the horizontal and the vertical direction by splitting the box into horizontal and vertical strips. The number of strips in the vertical and horizontal direction is determined based on the following formula:
   number of strips (vertical or horizontal) = $\lfloor$(length of the edge (vertical or horizontal) of the box$\div$ average transmission range) $*$ 4$\rfloor$
   This is an empirical formula (as are all formulas in this section) derived from experience. Intuitively, using few wide strips will lose distribution information while very thin strips tend to have very low frequency counts. This formula appears to give us a good separation.
   (b) Find the peaks and valleys in the bar graphs. Select consecutive valleys such that there are at least five strips in between (this is done to avoid selecting very small clusters and also to ensure that a cluster is at least as wide as the transmission radius). Use the center of the strip that represents a valley (in both the X and Y bar graphs) to divide the whole area into rectangular boxes. This forms the box list.

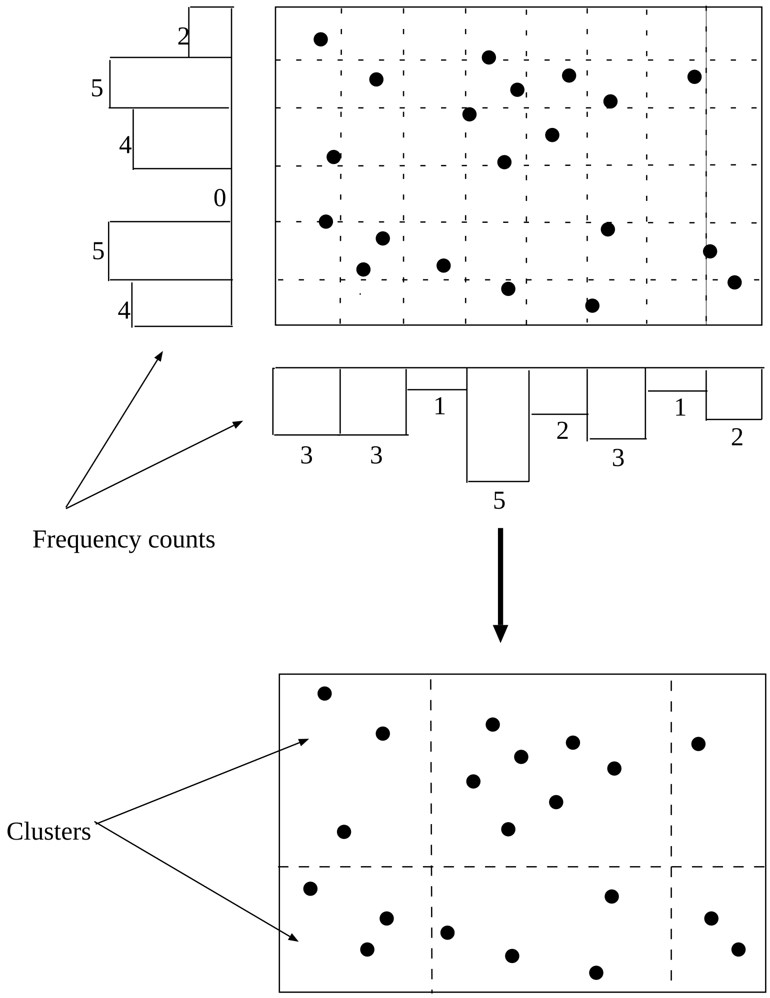


Figure 7: An example of the periodic clustering algorithm.

2. *Box Size Refinement*

   (a) We allow the maximum length of a box to be THREE times the transmission range. This is because we need to keep the overhead of data collection at the cluster head small. If a box is bigger along a dimension, we split it evenly along that dimension.

   (b) The maximum allowed ratio of the two edges of a box is 1.5. We call this the box edge ratio rule. Check all the boxes for edge ratio, if a box violates this rule, we split the box along the longer edge evenly. The rationale for this rule is to prevent the formation of long and skinny boxes.

3. *Node density adjustment*

   (a) Count the number of nodes in each box.

   (b) If a box is empty, delete this box from the box list.

   (c) Let the minimum and maximum allowed number of nodes in a box be MinN and MaxN. If a box B_i has less than MinN nodes, try to merge B_i with one of B_i's neighbors B_j according to the following merging steps.

      i. Find out B_i's left, right, bottom, and top neighbors, B_l, B_r, B_b, B_t, respectively, and put them into a candidate set S. Whenever S is empty, merging stops.

ii. Check node density for every box in the set S. If the number of nodes in B_i plus the number of nodes in a neighbor is larger than MaxN, remove the neighbor from S.

iii. Check box size restriction on every box in set S. For instance, if expanding B_r to contain B_i would cause B_r's edge to be longer than the maximum edge length or violate the edge ratio rule, then delete B_r from S.

iv. If there are more than one box left in S, choose the box B_j from S such that it can contain B_i with the smallest increase in edge length. If there is only one box left in S, this box is B_j.

v. Merge B_i with B_j. Move one of B_j's edge to contain B_i, all the nodes inside B_i are now inside B_j, delete B_i from the box list

(d) If a box B_i has more than MaxN nodes, try to split it according to the following splitting steps:

i. Let $P = B_i/MaxN$ and let E_x and E_y be the lengths of B_i's X and Y edges, R is the transmission range. First check if splitting B_i will result in a violation of the box edge rule (ratio of edges is no more than 1.5). If it does, stop the splitting procedure. If not, we consider 2 cases.

ii. *Case 1* $P \leq 2$: If $max(E_x, E_y) < R$, it makes no sense to split B_i because any two nodes inside B_i can reach each other. So only split B_i if $max(E_x, E_y) > R$. Split B_i into two equal boxes in the longer side.

iii. *Case 2* $P > 2$: If both E_x and E_y is larger than R, split B_i into 4 equal boxes. If only one of E_x and E_y is larger than R, split B_i into 2 equal boxes. If neither is larger than R, do not split the box.

(e) If a box B_i has only a single node in it, try to make that node a *guest* of some other box according to the guest protocol (discusses in the next subsection).

After clustering is complete, the manager selects a cluster head for each cluster based on locality (e.g., as close to the center as possible) or some other metric (e.g., maximum available battery power). It then multicasts the clustering information to these cluster heads.

5.2.2 Cluster Maintenance

This protocol runs in between the periodic invocations of the above algorithm. In order to better describe the workings of this protocol, let us consider the different situations that can occur:

1. *Node departs from a box:*

   - When N finds itself leaving the home box, it sends a disjoin message to the cluster head, and starts to look for a new home box. To do this, N broadcasts a message to all its neighbor nodes, telling them it is looking for a home box. One neighbor out of each box replies with the information of its home box including the box's geographical coordinates, the cluster head ID, and possibly the list of nodes belonging to this box.
   - Based on these responses, if N finds itself currently located inside a box, it sends a join message to the cluster head.
   - If N could not find a home box (i.e., it is not located in any box that has a cluster), it tries to find a *nearby box* and join it as a guest using the guest protocol (to be discussed later in this section).
   - If N could not find either a home box or a host box, it forms a new square box centered at its location with its transmission range as the edge length and itself as cluster head, and registers with the manager.
   - If a node does join a new box, the cluster head determines if the number of nodes in its cluster is greater than MaxN. If so, it *sets a box splitting flag* and waits for some time. After a timer goes off, if the number of nodes in the box has not changed, the cluster head splits the box (using the algorithm from the previous section). The reason for using a timer is to delay splitting in the hopes that some node may leave, thus dropping the node count to less than MaxN. This optimization saves a significant amount of message overhead.
   - The cluster head at the node's old box determines if the number of remaining nodes has fallen below MinN. If so, it sets a flag and if no changes take place in the membership after a timeout period, it attempts to merge with a neighboring box (again using the procedure outlined earlier).

2. *Cluster head departs from a box:* Say the cluster H moves out of box X. If the number of nodes in X is now zero, H informs the manager to delete the box. Otherwise, the H informs up to three nodes of X that it has departed (these nodes are selected based on how centrally they are located within the box X). One of these is then selected as cluster head based on a metric of choice (e.g., battery power remaining, slow relative speed, etc.).

3. *Guest protocol:* Sometimes a node is in a box all by itself but is connected to nodes in neighboring boxes that contain clusters. In such a case, rather than making the node form a cluster all by itself, or even merging the box containing the solitary node with a neighboring box, we treat the node as a *guests* in one of the neighboring clusters. The rule of thumb followed is simple. Say the node is connected to a nodes of the cluster C_a and b nodes of cluster C_b. Then the node becomes a guest of cluster C_a if $a > b$.

   In order to facilitate bookkeeping, we require guests to perform additional tasks.

   - Each node has a status flag indicating whether it is a guest.
   - If a guest node finds itself moving into its home box, it sends a join message to the cluster head and cancels its guest status.
   - When a guest hears a disjoin message from or loses connection with any node in C_a that it was connected to, it decrements a. When $a = 0$, the node begins searching for a new home.
   - For cluster heads to be able to keep track of their guests, guests are required to send a guest beacon once in while. If a guest fails to do that, the cluster head removes the guest from the guest list. This is done by having a timer at both the guest and the cluster head.

4. *Cluster head ping protocol:* It is sometimes possible that a cluster head may be disconnected abruptly from its cluster (either due to fading, jamming or other catastrophic failure). In such a case, the cluster nodes are left unmanaged and *they do not know that they are unmanaged.* The solution is to require cluster heads to periodically broadcast a ping messages. Nodes maintain a timer that is reset whenever a ping is received. If the timer goes off, the nodes begin the search for a new cluster head.

Let us consider an example of protocol operation as illustrated in Figure 8. In Figure 8(a) we have a situation where node 1 is a guest of cluster C5. Figure 8(b) illustrates what happens when node 9 (the cluster head of C2 and the network manager) moves out of its cluster and into C4. Node 2 is selected as the new cluster head in C2. Node 11 continues to be the cluster head of C4 even though node 9 (the manager) moves into its cluster. Figure 8(c) shows the evolution of the network when nodes 6 and 11 move out of their cluster C4. When this happens, clusters C4 and C5 merge to form C6. This figure also shows that node 8 has moved out of C5. It becomes a guest of cluster C6. In Figure 8(d) we see that clusters nodes 2 and 8 move into C3 while node 3 moves into C6 thus C2 disappears. Finally, cluster C6 has too many nodes and is split into two in Figure 8(e).

5.2.3 Performance Results of Geographical Clustering

We use the same simulation methodology as that used in section 5.1.2. We examine the tradeoff in message overhead versus the percent of nodes unmanaged by clusterheads. The message overhead comes about in three ways:

- The ping messages sent by the cluster head to inform nodes in its cluster that it is alive,
- The messages exchanged to maintain the clusters, and,
- the messages sent by the manager each time the centralized procedure is run.

In the plot on the left in Figure 9 we graph the message overhead as a function of node speed for different ping intervals. As we can see, the message overhead increases with smaller ping intervals. However, we observe a significant drop in the percentage of nodes unmanaged by clusterheads when the ping intervals are frequent. In this set of plots, the periodic clustering algorithm is run every 30 time steps and a guest beacon is generated every 3 time steps. In Figure 10 we plot the message overhead and the percentage of nodes unmanaged by clusterheads for the case when the guest beacon is generated every 3 time steps and the manager sends pings every 15 time steps. Different curves represent different intervals when the periodic centralized clustering algorithm is run. It is interesting to observe that at high speeds, the message overhead for c-intvl = 30 (representing frequent invocations of the centralized clustering procedure) is actually lower than for c-intvl = 500. This is because at high speeds, the maintenance algorithm experiences a huge increase in message overhead. It is thus beneficial to run the centralized clustering procedure more frequently. We see that the percentage of nodes unmanaged by clusterheads also falls with more frequent centralized clustering.

Figure 11 also plots the message overhead and number of nodes unmanaged by clusterheads for the case when the ad hoc network contains 90 nodes. As we can see, the behavior of the algorithms for this case is similar to that of the 30 node case. Finally, Figure 12 plots the number of messages per second versus speed for different network sizes ranging from 30 nodes to 150 nodes. The ping interval is 15s, c-intvl is 30 and the guest beacon is generated every 3s. It is interesting to observe that the message cost increases linearly with speed and with number of nodes. Thus, our algorithm scales well.

5.2.4 Comparison of Graphical Clustering and Geographical Clustering

It is interesting to compare and contrast the behavior of the two clustering algorithms presented in this paper. In general, we see that the message cost of maintaining clusters (at all speeds) is slightly smaller for graphical clustering. However, we note that the percentage of nodes unmanaged by clusterheads for graphical clustering, when no MAC information is available, is very high. Thus, in this situation, geographical clustering may be the better choice because the percentage of nodes unmanaged by clusterheads remains well below 10% for all speeds and network sizes. If we have a situation where nodes have widely varying transmission ranges, however, graphical clustering is better because geographical clustering does not consider this information.

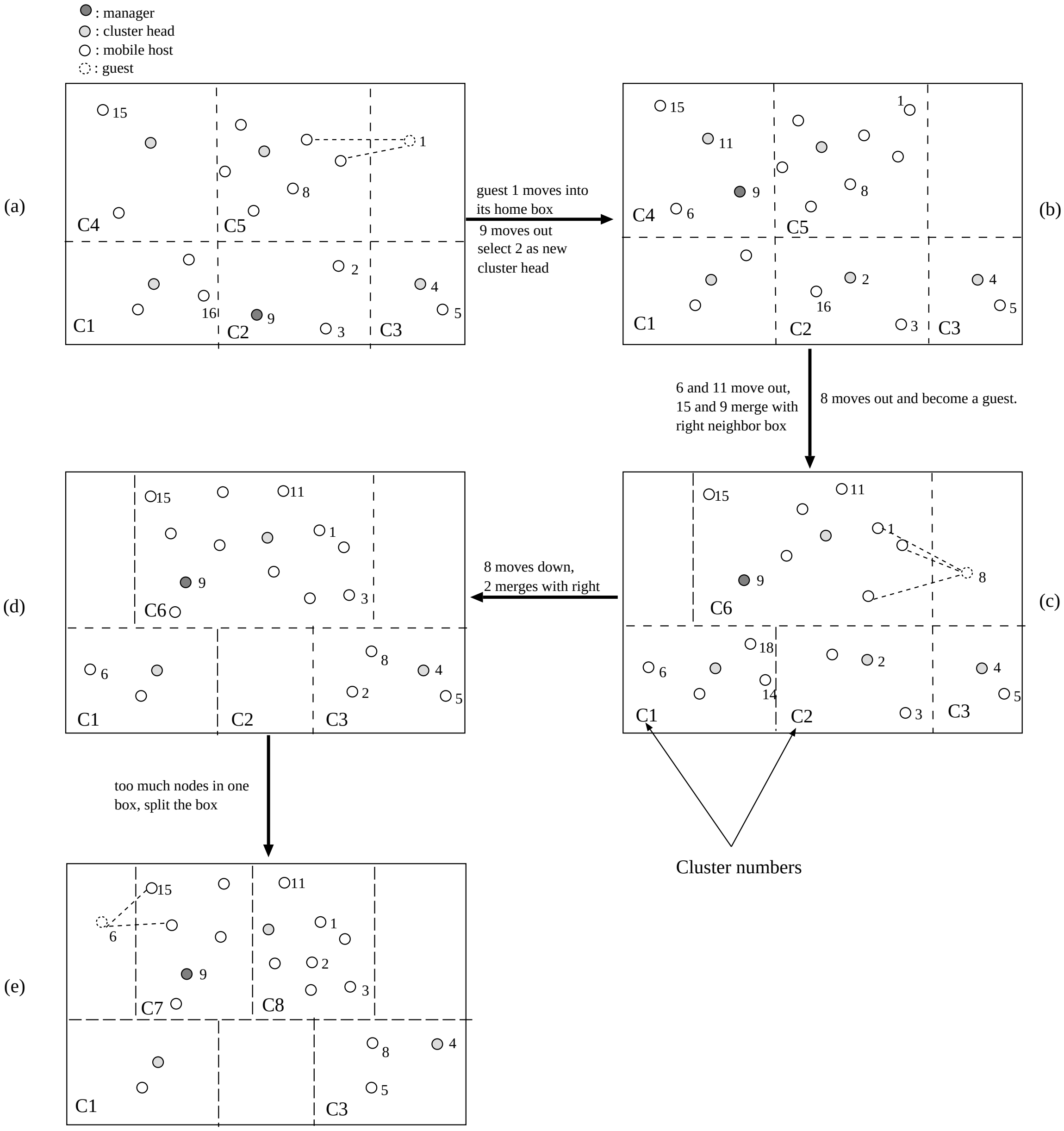


Figure 8: Maintenance component of geographical clustering.

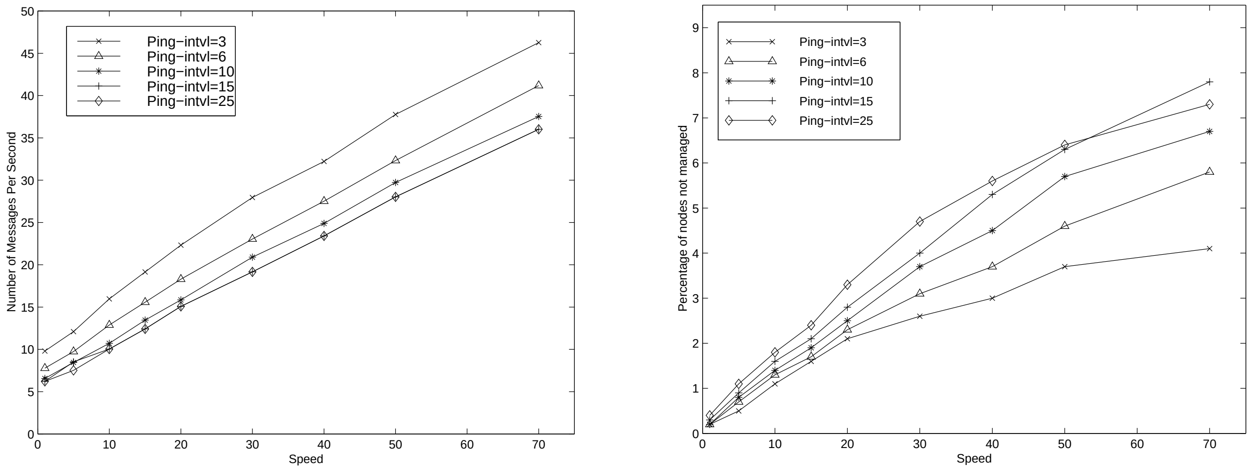


Figure 9: Message overhead and Percentage of nodes unmanaged by clusterheads (periodic clustering run every 30 time steps, guest beacon period = 3, Nodes = 30).

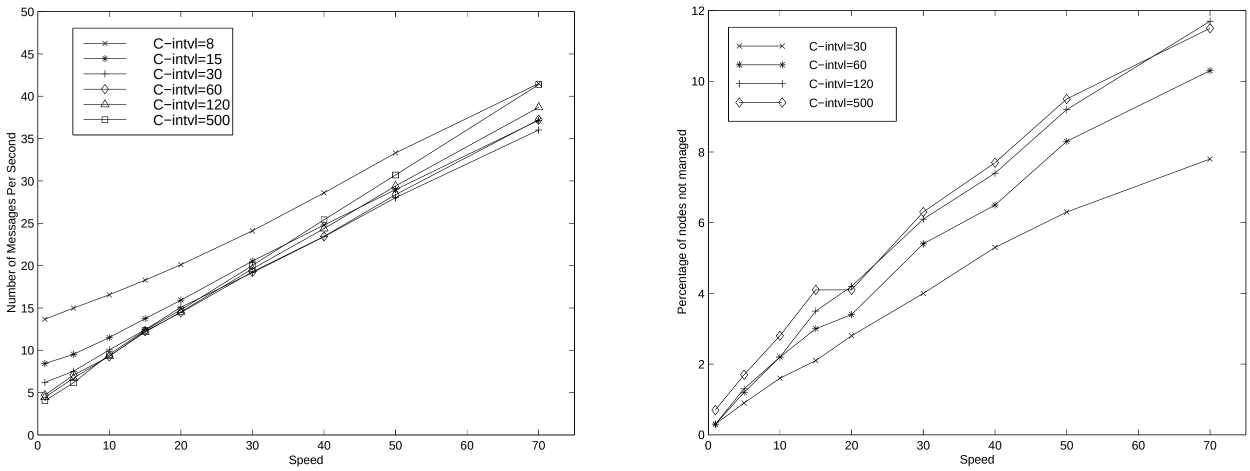


Figure 10: Message Overhead and Percentage of nodes unmanaged by clusterheads (guest beacon period = 3, ping interval = 15, Nodes = 30).

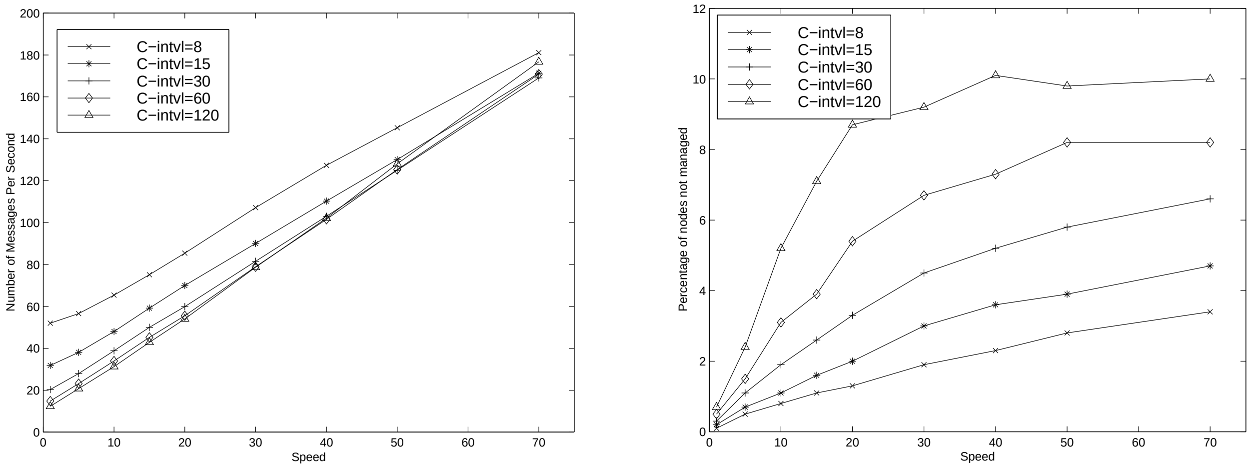


Figure 11: Message Overhead and Percentage of nodes unmanaged by clusterheads (guest beacon period = 3, ping interval = 15, Nodes = 90).

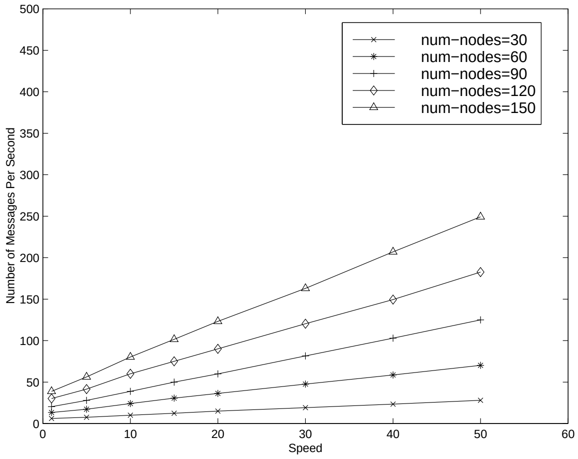


Figure 12: Message overhead (ping = 15, c-intvl = 30, guest beacon = 3).

6 Security in ANMP

ANMP implements sophisticated security mechanisms that allows it to be usable in mission critical environments. In sections 2.1.3 and 4.3 we presented scenarios demonstrating the need for different types of security in ad hoc environments. In this section, we discuss the details of the security sub-system in ANMP and indicate how ANMP meets the security requirements we have discussed.

6.1 Secure Multicasting

Secure multicast is needed for the management of ad hoc networks to allow the manager to configure networking elements safely and to ensure secret distribution of information between cluster heads and the manager. The properties of a good secure multicasting protocol are:

1. Message efficient,
2. Maintain secrecy from eavesdroppers,
3. Protect against replay attacks where some node, not in the multicast group, retransmits a previous multicast,
4. Protect against insider attacks where a compromised multicast group member attempts to tamper with the contents of a multicast message that it is forwarding down the multicast tree,
5. Protect against authentication failures where a compromised multicast group member initiates a completely fictitious multicast session.

We utilize the secure broadcasting protocol of [10] as the basis for ANMP's secure multicast. In this protocol, the manager multicasts a message of the form $\boxed{\text{X, CKD, C}}$ to all nodes of the multicast group. Since the same message is multicast to all group members, this protocol satisfies the first requirement of message efficiency mentioned above.

Let us now look at the specific components of the above message. Let M be the plaintext message that needs to be multicast. Let (s_e, s_d) denote the *session* encryption key and decryption key respectively. The message M is encrypted using s_e and is decrypted by all receivers using s_d. All members of the multicast group have their own encryption and decryption keys $(e_i, d_i) \forall i \in G$ where G is the multicast group. The components of the message are:

$$\begin{aligned} C &= E_{s_e}(M) \\ \text{CKD} &= E_{s_e}(s_d) \end{aligned} \tag{1}$$

The message M can be decrypted by all group members if they have the decryption key s_d. It is impractical to transmit s_d separately to each member of G, however, since the keys (s_e, s_d) change with each message. An interesting way to *multicast* s_d securely was developed by [10]. This technique is based on the Chinese Remainder Theorem and works as follows. For each $i \in G$, the sender computes,

$$R_i = E_{e_i}(s_d)$$

Then the sender computes X such that,

$$X \equiv R_i \bmod N_i, \forall i \in G \tag{2}$$

where N_i are pairwise relatively prime integers and are publicly known to all. In order to retrieve R_i, each receiver solves equation 2, then decrypts R_i to obtain s_d after which M is obtained. The receiver also verifies that $D_{s_d}(\text{CKD}) = s_d$ to prevent impersonation by outsiders.

The security of the above protocol is, unfortunately, not always guaranteed. Table 4 illustrates the potential problems for different combinations of the keys (s_e, s_d) and (e_i, d_i). Specifically,

1. If we use asymmetric encryption between nodes (i.e., $e_i \neq d_i$), then it possible for a compromised group member g to *forge* an entire multicast because e_is are publicly known to all. That is, g can generate a fake message M' and fake session keys (s'_e, s'_d). g computes $R'_i = E_{e_i}(s'_d)$ and generates a perfectly legitimate looking message. There is no way for a receiver to know that this session was a fake. We refer to this problem as an *authentication failure.*

	Asymmetric Session Key $s_e \neq s_d$	Symmetric Session Key $s_e = s_d$
Group Member Asymmetric Encryption Key $e_i \neq d_i$	Authentication Failure	1. Authentication Failure 2. Insider Attack
Group Member Symmetric Encryption Key $e_i = d_i$	Safe	Insider Attack

Table 4: Problems with the basic multicast protocol.

2. If we use asymmetric (e_i, d_i) and symmetric keys for the message (i.e., $s_e = s_d$) then in addition to the authentication failure, we have the potential for an *insider attack*. That is, a compromised group member g can modify M of an ongoing multicast. This is easy to do because g knows the key s_e (which is equal to s_d). Thus, g can modify M, and change C and CKD appropriately.

3. If we use symmetric encryption for the session ($s_e = s_d$) and for all group members ($e_i = d_i$) then we again have the problem of insider attack as discussed above.

The solution to these problems is to use a message digest and incorporate that as part of the R_is. Thus, in ANMP we use,

$$R_i = E_{e_i}(s_d, D_{k_r}(\text{Message digest, Session ID, Timestamp}))$$

where (k_p, k_r) are a pair of public and private keys maintained by the sender. k_p is known to all but k_r is secret. To see why this method works, observe that no one except the legitimate sender can compute E_{k_r}. On receiving a multicast, a receiver can extract the message digest (computed using MD5, see [9]), the session ID and the timestamp (to prevent replay attacks) correctly by computing,

$$E_{k_p}(D_{k_r}(\text{Message digest, Session ID, Timestamp}))$$

6.2 Level-based Access Control Model (LACM)

ANMP implements the military security model that meets the requirements described in section 2.1.3. Essentially, we think of each node as having a *clearance level* and each data item having a *security level*. If a data item has a security level of 3 (small numbers denote higher security levels) then only nodes with clearance of 0, 1, 2 or 3 can read this data item. A generalization of this model is to group data items and nodes into *compartments* (we can think of compartments as individual projects) where a node may have different clearance levels in different compartments (e.g., a person working on project A has no need to know details of project B even though that person has a high clearance level for project A). In ANMP, all nodes in the ad hoc network are assigned a set of tuples of the form,

$$S^j = \{(c_1, l_1), (c_2, l_2), \cdots, (c_k, l_k)\}$$

where c_i denotes a *compartment* and l_i denotes the clearance level node i has for compartment c_i. There are k compartments in all defined for the ad hoc network (this number is dynamically extensible by the manager). S^j describes the clearance level node j has for all compartments. The manager issues a *certificate* to j containing S^j. These certificates cannot be forged but they can be read and verified by all nodes (see [9] for a description of various techniques of issuing certificates).

When an agent is queried for data by the cluster head or by the manager, the agent first determines the security clearance level of the cluster head. If the cluster head has a lower clearance level, the agent encrypts the data such that only the manager can decrypt it. If, on the other hand, the cluster head has a higher or equal clearance level, the data is encrypted so that the cluster head can decrypt it. When the cluster head prepares a summary of the data for the manager, it assigns the summary a clearance level equal to the highest clearance level of the data that was summarized.

We discussed the need for information exchange between managers of co-located networks in section 2.1.3. In our LACM model, we allow a user to assign security classification levels to managers in the form of certificates. Managers then respond to information requests from other managers by appropriately editing and/or summarizing the data. As an example, consider a disaster scenario where an ad hoc network of relief personnel is co-located with an ad hoc network of news organizations. The manager of the relief network may respond to the manager of the news network's queries about the extent of damage by providing a casualty count rather than the names of the individuals. We can imagine many similar applications of this model in a battlefield environment as well.

6.2.1 Effect of LACM on Clustering

It is easy to see that security information must be brought to bear upon the process of cluster head selection. In other words, if a cluster head has a very low security clearance (as compared with the agents in its cluster) then the cluster head may be unable to manage the cluster efficiently for the following reasons:

1. Positional information of nodes is very sensitive data and needs to be protected. Thus, in military scenarios, it is possible that this information will be assigned a high security level. This is also true of information such as the remaining lifetime of a node, etc. If a cluster head has a lower clearance level than the security level of this information, then *it cannot track nodes within its cluster*!

2. If the cluster head has a low clearance level, then it is likely that most information collected from the nodes will be encrypted such that only the manager can decrypt it. As a result, the savings in message overhead we hoped for is lost (the cluster head is unable to prepare summaries since the data is encrypted).

A solution to this problem is to ensure that the cluster head is selected in a way that it has a high security clearance level. The graphical clustering protocol can be very easily adapted to do this (cluster heads can be selected based on security clearance levels instead of minimum IDs). In the case of the geographical-based clustering approach, on the other hand, we need to modify our central periodic clustering algorithm (see section 5.2.1) to form clusters based on the density as well as the security levels of nodes (we discuss this simple modification further in [26]).

7 Data Collection via MIBs

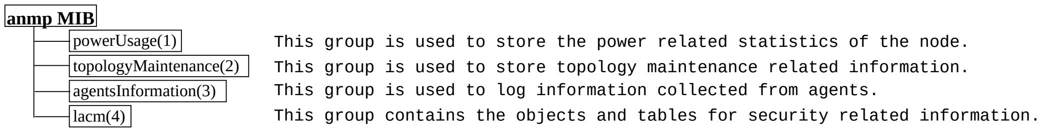


Figure 13: Ad Hoc Network Management MIB.

Every node in the network runs an ANMP entity locally. To support network management functions, we add a new MIB group, called *anmpMIB*, to the standard MIB-II of SNMP. This group defines objects that reflect the characteristics specific to ad hoc networks. Figure 13 lists the subgroups, which are in anmpMIB. The first group is the *powerUsage* group which group defines objects related to battery power and power consumption statistics for all the interfaces of the node. Figure 15 lists these objects most of which are self-explanatory. An interesting object here is the *powerBatteryDrainFunction* entry that describes the drain function of the battery used in the mobile. An example of such a function is illustrated in Figure 14 which plots the measured voltage against remaining life for Lithium-Ion batteries. To determine the remaining life, the mobile only needs to measure the voltage of the battery and then use the drain function data. When the mobile puts in a *new* battery, the manager can download (to the mobile) the drain function corresponding to this new battery. Other information stored in the powerUsage group is the energy expended in transmitting and receiving 32-bytes (these are the powerTransmitPower and powerReceivePower entries). This information is useful in determining energy-efficient routes for transmitting packets (see [19]).

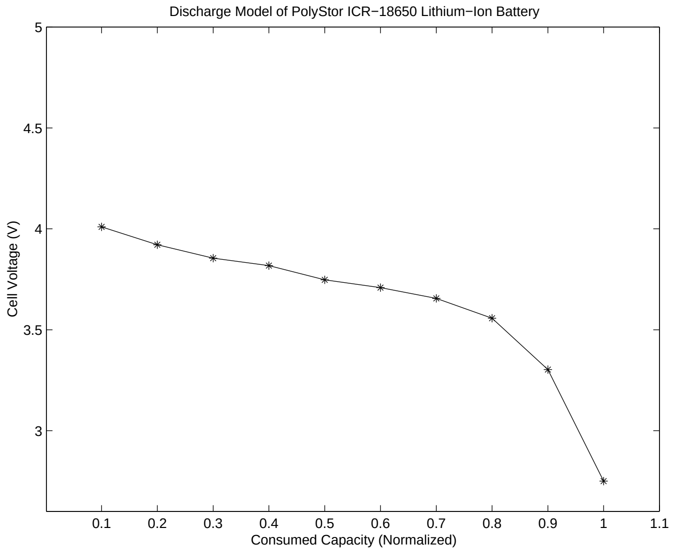


Figure 14: Discharge curve for a Lithium-Ion battery.

The second group is the *topologyMaintenance* group. This group (see Figure 16) defines objects related to the topology information of the ad hoc network. The neighborTable contains a list of all one-hop neighbors of a node. The protocol table contains one entry for each protocol, which may be used for topology maintenance. Currently, we have two protocols in the topology maintenance group – graphical clustering and geographical clustering. New entries for other topology maintenance algorithms can be added here. The graphicalClustering group is made up of objects that are used by the clustering algorithm described earlier. Figure 17 lists the objects in *graphicalClustering* group. Each node stores its cluster information in the cluster table. The *geographicalClustering* group is illustrated in Figure 18. The entries here are self-explanatory. Figure 19 is the MIB for the *mobileHosts* group, which is used by the geographical clustering algorithm.

The third group in anmpMIB is the *agentInformation group.* The manager and cluster heads use the tables in this group to store information collected from the agents. Figure 20 lists some of the subgroups, which are part of

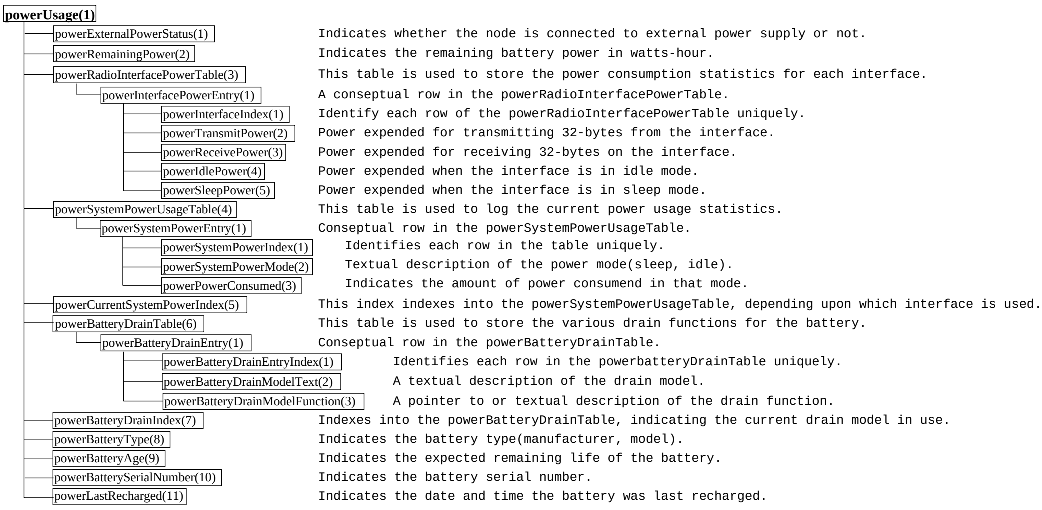


Figure 15: Power Usage group.

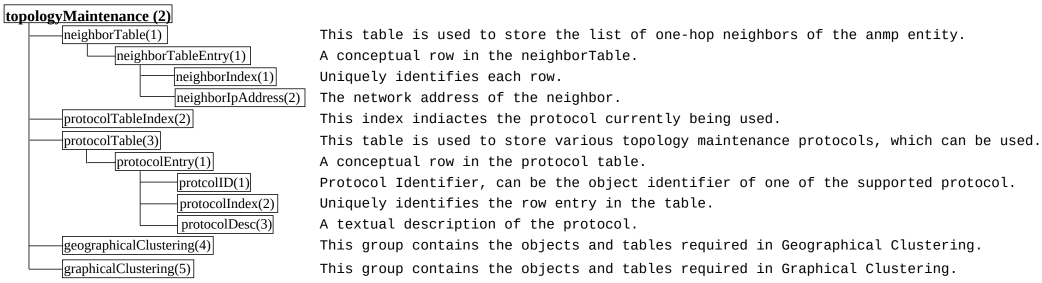


Figure 16: Topology Maintenance group.

this group. It may be noted that the agents information group can be very exhaustive to include all possible objects in the MIB. We elaborate on one subgroup here, Figure 21. More subgroups can be added, as desired, along similar lines.

The *informationUpdateInterval, percentageAgentManaged, percentageAgentUpdateInfo, mobileGatewayFlag* and *mobileGatewayStatus* in Figure 20 are scalar objects. The informationUpdateInterval, in millisecs, dictates the polling interval for the cluster head and manager. The object percentageAgentManaged shows the percentage of agents a node is responsible for. The percentageAgentUpdateInfo gives the percentage of nodes in a cluster for which the cluster head has updated information. The mobileGatewayFlag dictates whether a node can act as a gateway or not (e.g., as a cluster head in the clustering algorithms) and the mobileGatewayStatus object shows if the node is currently a gateway or not.

Except for the *alarm* and *event* group all the subgroups in agentsInformation group are used to log data. The design of these other groups has been borrowed from the RMON-MIB[18]. Each subgroup has a control table. Associated with each entry in the control table is a data table that is used to store information. The use of control tables allows the manager to dynamically configure the cluster head to collect the desired information from all the nodes in its cluster. To do so the manager creates an entry in the control table of the desired subgroup. Along with other details, the manager can specify the MIB variable on which it wants samples and the preferable sampling interval. Depending upon the control table entry the cluster head starts logging information in the associated datatable.

The *alarm* and *event* subgroups in the agentsInformation group are used to facilitate manager to manager communication. This group has been adopted from the M2M-MIB[18]. The manager can configure a cluster head to unsolicitately send information. This can be done be setting up alarms and events in the alarm and events groups respectively. To do so, the manager creates a row in the event table. An event is associated with a row in the alarm table. A rising or falling threshold value can be specified on which the alarm is to be generated. Thus, whenever an

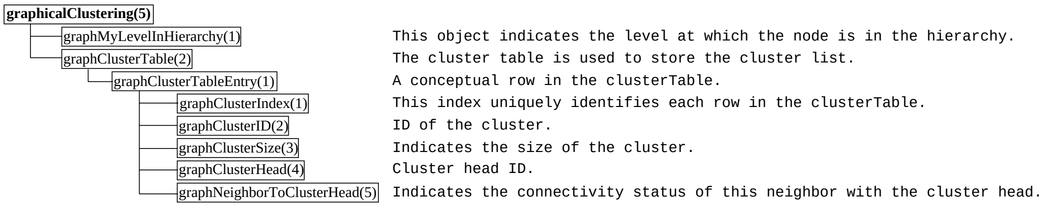


Figure 17: Graphical Clustering group.

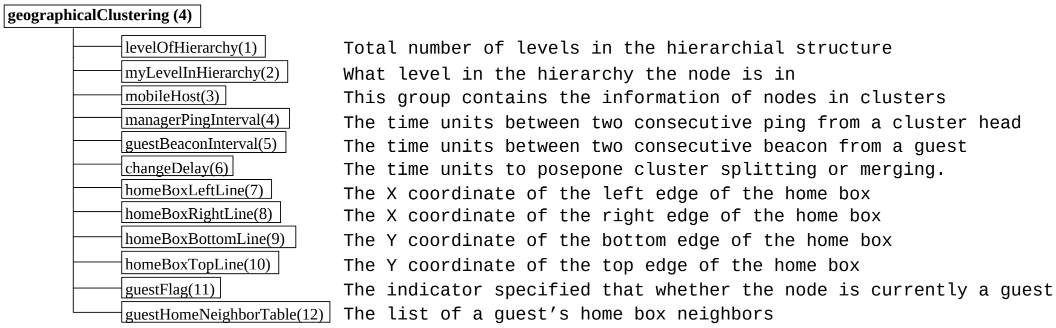


Figure 18: Geographical Clustering group.

alarm is triggered, it generates an event notification and sends it the destination. The alarm and events subgroups provide the manager with the capability to dynamically alter the network sampling statistics on which it wants to receive notification. This kind of unsolicited information exchange can result is a substantial bandwidth saving. To enhance the functionality of the event subgroup, we propose that an agent should be able to download and link a new function to its code, thus making the SNMP agent dynamically extensible. Since SNMP does not provide such a feature, we set the *eventsFunctionName,* in the events group, using SNMP PDUs. Then by some other means, like FTP, the agent downloads the function code and link it to its code. Such a feature enhances the operation of the event subgroup from just notification to action. Using a similar feature the MIB can also be dynamically extended. Figures 22 and 23 show the events and alarm groups respectively.

The last group which we add to anmpMIB is the *LACM* group (see Figure 24) consists of the information of classification and the security clearance of mobile hosts. There are two tables in this group, one is *objectSecurityLevelTable*, the other is *agentSecurityRankTable.* Each row in the objectSecurityLevelTable contains classification data about the corresponding object. The table is indexed by objectID as well as by the table index. This table contains the following attributes:

- objectID: the unique identifier of the object, its value is a sequence of integer.
- objectSecurityLevelIndex: a unique value for each row in the objectSecurityLevelTable.
- objectSecurityLevel: the security level of the object specified by the manager.
- objectProject1-6: the projects which the object is associated with.

A similar structure is used for holding the security clearance information about mobile hosts in agent SecurityRankTable.

From the above discussion it can be seen that our network management protocol has a three-level hierarchy. The top level is the manager who manages the entire network. At the second level are cluster heads and the agents are at level three. Since the topology changes are very rapid in an ad hoc network, trap directed polling is a better way to collect information. The manager station can use the alarm and event groups to configure the cluster heads to send information to the manager. The cluster head can use similar mechanisms to collect information from its agents. A summary of this information is then sent to the managers.

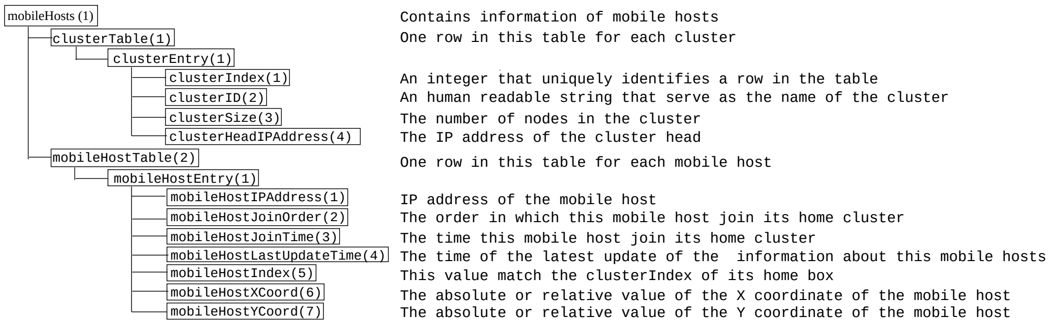


Figure 19: mobileHosts group.

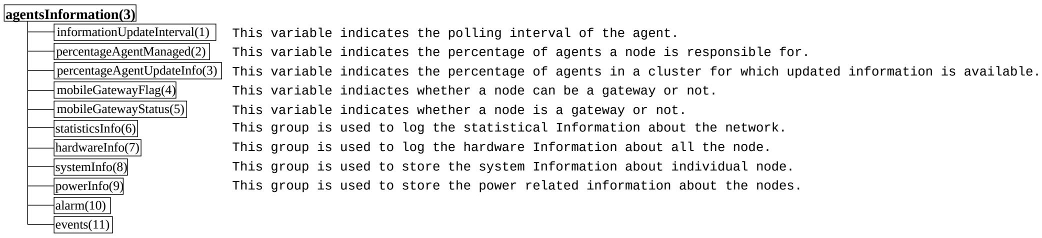


Figure 20: Agent Information group.

8 ANMP User Interface

No matter how efficient network management system is, its true potential cannot be realized if there is no effective mechanism to present the collected information to an end-user. This information can be presented to the human manager in the form of logs and reports. As the amount of information increases, however, this form of presentation has its own limitations. Therefore, a more efficient way to present information is an interactive graphical representation.

To provide the human manager with an effective tool to accomplish the complex task of managing the network, we have developed a GUI for the management station. The interface has been written in Java version1.1 using the Java Development Kit for Linux. We chose to write the code in Java because it makes the code portable. Java also has a rich collection of classes to support various GUI features and applications can be developed relatively quickly.

Our interface provides a graphical representation of the network topology. Each node in the network is represented as a dial, which indicates its remaining battery power. The color of the node depends upon the display criteria selected to view the network. In a drop down menu, we provide selectable criteria for the manager to view the network. For example, in the options menu, if the manager checks the protocol check box, all future refreshes of the network topology will display the nodes in the network indicating the protocol they are running. By default, the network topology is displayed with an indication of the cluster heads. Figure 25 shows the Management Window.

In the interface, we provide the manager with the facility to create SNMP PDUs using the *Query Box*. The manager can create Get requests and Set requests PDUs. The node ID from the graph is mapped to its network address internally. Figure 26 shows the query dialog box. In creating queries the manager can either type in the SNMP variable or use the browse option to select its variable from a drop down list, see Figure 27.

In addition to individual queries, we also provide a facility to view a node's information by clicking on the node. Figure 28 shows a sample screen of the fact sheet. In writing this interface application, we have used Java's *Abstract Window Toolkit (AWT)*. We have extended the *Frame* and *Dialog* classes of AWT to create the graphical interface. To provide functionality to the interface, we implemented *ActionListener* on various components and *MouseListener* to trap the clicks. To create a customized menu, we used the *MenuBar* class.

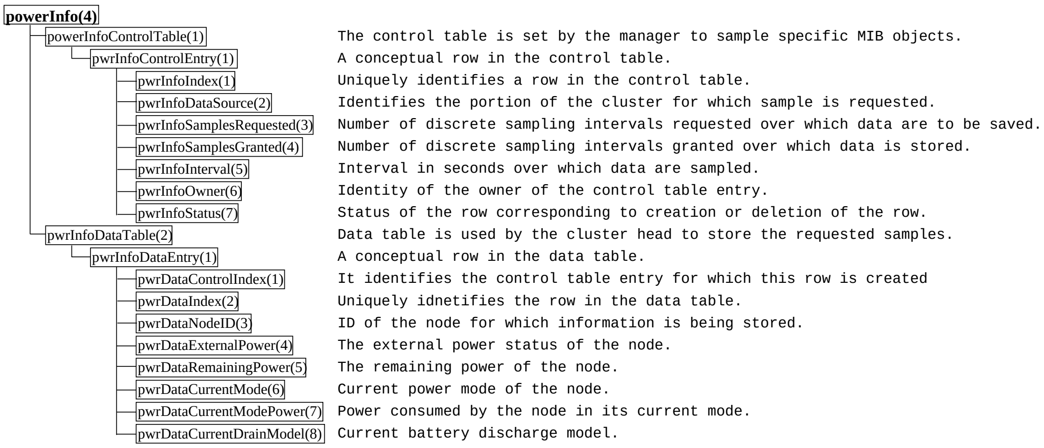


Figure 21: Power Information group.

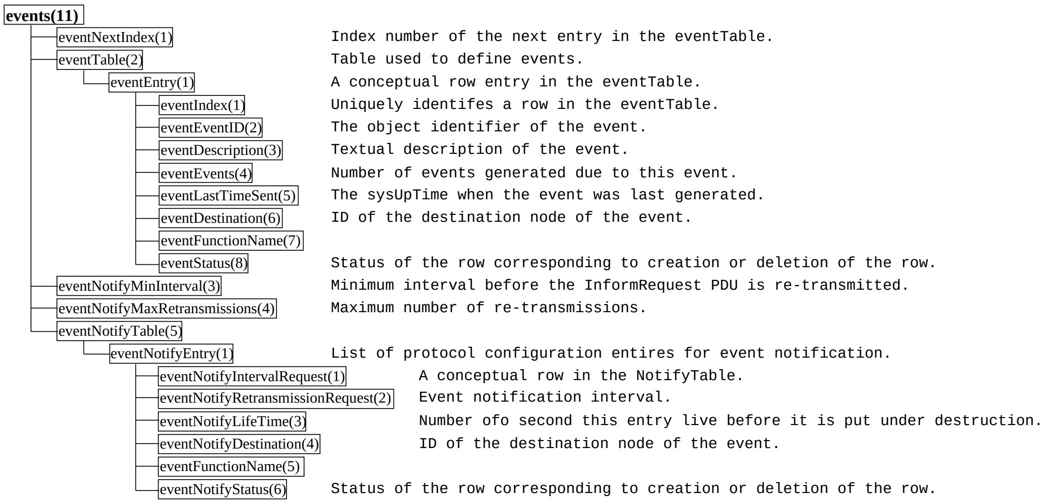


Figure 22: Events group.

9 Conclusion

Efficient Network management of ad hoc networks cannot be achieved using the existing protocols without significant functional enhancements. In this paper, we have presented our protocol, ANMP, that is based on SNMP. The significant differences include: MIB extensions, dynamic configuration of agents, dynamic extension of the agents and an application specific security module. We have also developed a user interface for the manager station.

To support distributed management, we have proposed an efficient clustering algorithm. A unique feature of our algorithm is that we are able to differentiate between node movements inside the cluster and across the cluster boundary, no messages are exchanged between clusters. It may be noted that in both the schemes proposed, we can ensure that well over 90% of nodes are managed while keeping the message overhead as low as possible.

Since there has not been much research on management protocols for wireless networks, there is a great deal of scope for further research in this area. It would be interesting to explore other ways of clustering or methods to group the nodes in the network. Also, the user interface can be enhanced to provide more advanced network management functions and network configuration facilities.

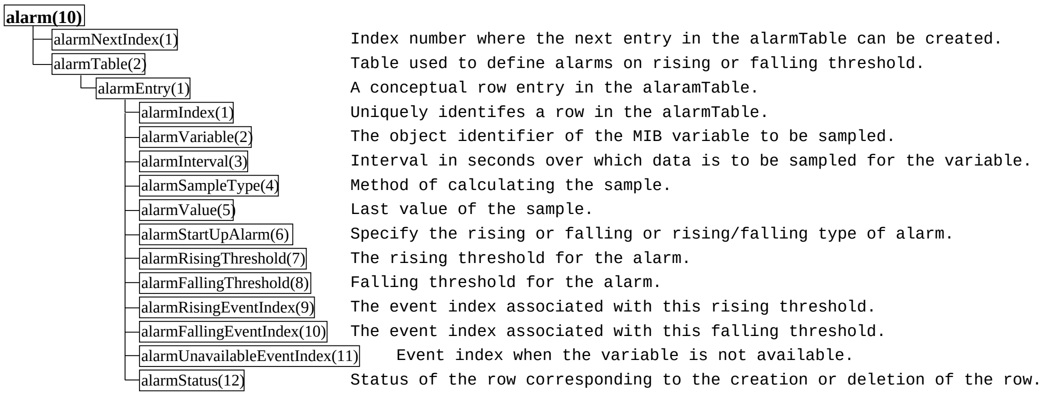


Figure 23: Alarm group.

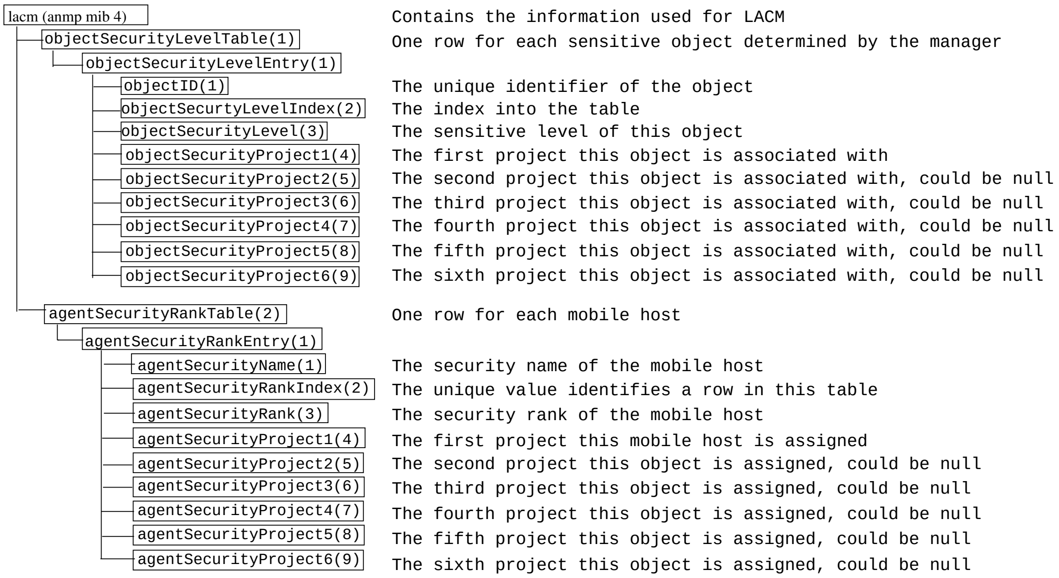


Figure 24: LACM group.

References

[1] K. Terplan. Communication Network Management, Prentice-Hall, Englewood Cliffs, N.J., 1989.

[2] Mobile MIB Taskforce, http://www.epilogue.com/mmtf/

[3] Juan Pavón, José Tomás. "CORBA for Network and Service Management in the TINA Framework". *IEEE Communications*, Vol. 36, No. 3, March 1998, pp. 72-79.

[4] Germán Goldszmidt, Yechiam Yemini. "Delegated Agents for Network Management". *IEEE Communications*, Vol. 36, No. 3, March 1998, pp. 66-70.

[5] Víctor S. Acosta. "OSF/DME (Distributed Management Environment)", http://www.frontiernet.net/ vsa184/papers/osf_dme.htm, April 1998.

[6] OMG. "CORBA Overview", http://www.infosys.tuwien.ac.at/Research/Corba/OMG/arch2.htm#446864, April 1998.

[7] U. Blumenthal, B. Wijnen. *User-based Security Model (USM) for version 3 of the Simple Network Management Protocol (SNMPv3)*. RFC 2274, January 1998.

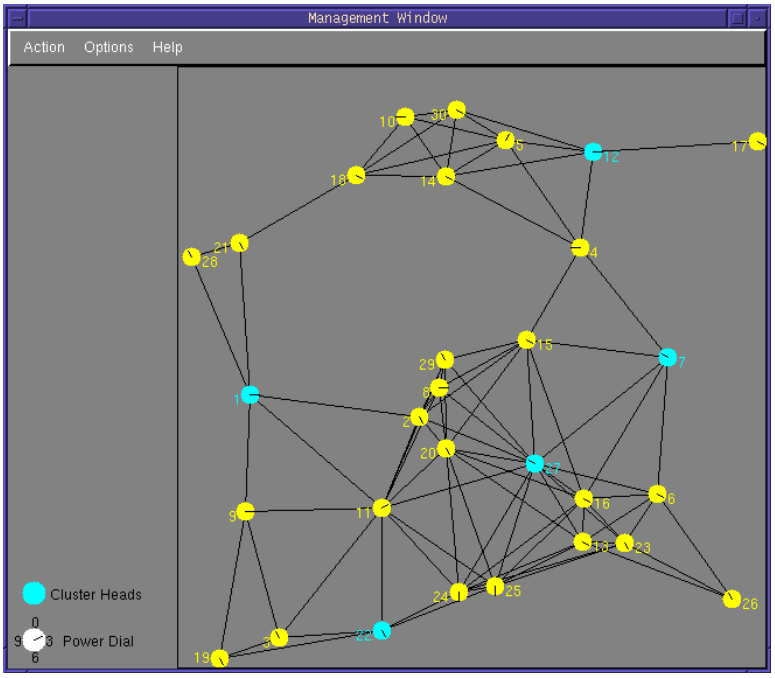


Figure 25: Management Window.

[8] B. Wijnen, R. Presuhn, K.McCloghrie *View-based Access Control for the Simple Network Management Protocol (SNMP).* RFC 2275, January 1998.

[9] Charles P. Pfleeger. Security in Computing, second edition. Prentice-Hall, 1997.

[10] Guang-Huei Chiou, Wen-Tsuen Chen. "Secure Broadcasting Using Secure Lock", *IEEE Transaction on software Engineering*, Vol. 15, No. 8, August 1989, pp.929-933.

[11] Li Gong, Nachum Shacham. "Multicast security and its extension to a mobile environment", *Wireless Networks*, Vol. 1 No. 2, 1995, pp 281-295

[12] John McLean. "The Specification an Modeling of Computer Security". *IEEE Computer*, Vol. 3, No. 1, January 1990, pp. 9-17.

[13] Merkle, R. "Protocols for Public Key cryptosystems". *Proc IEEE Symp Security and Privacy*, 1980, pp. 122-133.

[14] Kohnfelder, L. "Towards a Practical Public-Key Cryptosystem". MIT EE Bachelor's Thesis, 1978.

[15] Nitin Jain. "Graphical Based Ad hoc Network Management Protocol," MS Thesis, Department of Computer Science, University of South Carolina, Columbia, May, 1998.

[16] http://www.navcen.uscg.mil/faq/gpsfaq.htm, date April 1998.

[17] Allen Leinwand and Karen Fang, Network Management A Practical Perspective, Addison Wesley, 1993.

[18] William Stallings, SNMP, SNMPv2, and CMIP The practical guide to network management standards, Ist edition, 1993 Addison-Wesley Publication Company, Inc.

[19] C.S.Raghavendra and Suresh Singh, "PAMAS - Power Aware Multi-Access protocol with Signalling for Ad Hoc Networks", Technical Report TR-9801, Department of Computer Science, University of South Carolina, Columbia, SC 29208, March 24, 1998.

[20] William Stallings, "SNMP and SNMPv2: The Infrastructure for Network Management," *IEEE Communications Magazine*, Vol. 36, No. 3, March 1998, pp. 37-43.

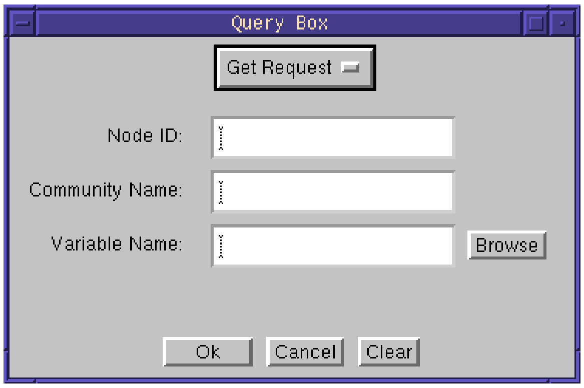


Figure 26: Query Box.

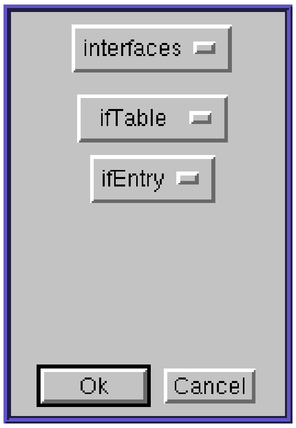


Figure 27: List Screen.

[21] *Network Mangement Server*, http://netman.cit.buffalo.edu/index.html, April 1998.

[22] U. Blumenthal, B. Wijnen. *User-based Security Model (USM) for version 3 of the Simple Network Management Protocol (SNMPv3)*, RFC 2274, January 1998.

[23] B. Wijnen, R. Presuhn, K.McCloghrie *View-based Access Control for the Simple Network Management Protocol (SNMP)*, RFC 2275, January 1998.

[24] Lin Chunhung R. and M. Gerla, "Adaptive Clustering for Mobile Wireless Networks," *IEEE Journal on selected areas in Communications*, Vol. 15, No. 7, Sept. 1997, pp. 1265-1274.

[25] M. Gerla and Tsai J. Tzu-Chich, "Multicluster, mobile, multimedia radio network," *ACM-Baltzer J. Wireless Networks*, Vol. 1, No. 3, October 1995, pp. 255-266.

[26] Wenli Chen, "Geographic based Management Protocol for Ad Hoc Networks," MS Thesis, Department of Computer Science, University of South Carolina, Columbia, May, 1998.

[27] SNMP Research Group, *The EMANATE Run-time Extensible Agent System*, http://www.snmp.com/emanateintro.html, April 1998.

[28] *SNMP Version3 Charter*,http://www.ieft.org/html.charters/snmpv3-charter.html, April 1998.

[29] Lakshmi Raman, "OSI Systems and Network Management," *IEEE Communications Magazines*, Vol. 36, No. 3, March 1998, pp. 46-53.

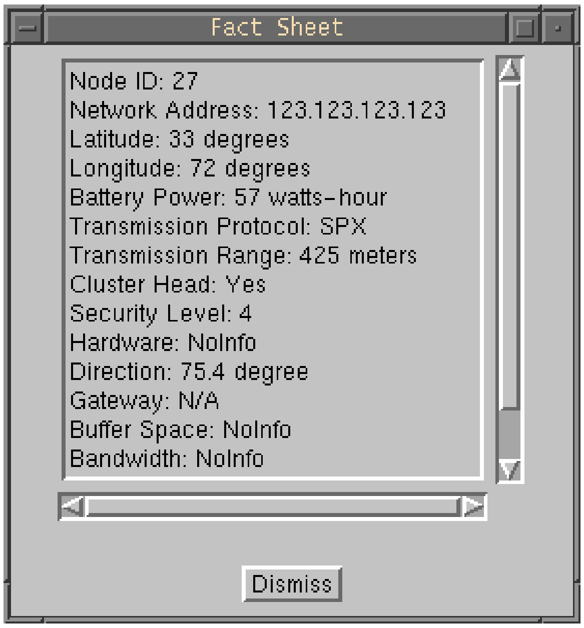


Figure 28: Fact Sheet.

[30] David J. Sidor, "TMN Standards: Satisfying Toda's Needs While Preparing for Tomorrow," *IEEE Communications Mangazine*, Vol. 36, No. 3, March 1998, pp. 54-64.

[31] Mohsen Kahani and H.W. Peter Beadle, "Decentralised Approach for Network Management," *ACM SIGCOM Computer Communication Review*, January 1997, pp. 36-47.

[32] *Mobile Management Task Force* , http://www.epilogue.com/mmtf/, April'1998.

[33] Z. J. Haas and M.R. Pearlman, "The Zone Routing Protocol (ZRP) for Ad Hoc Networks", *IETF MANET*, Internet Draft, Dec. 1997.

[34] David B. Johnson and David A. Maltz. "Dynamic source routing in ad hoc wireless networks", In Tomasz Imielinski and Henry F. Korth, editors, Mobile Computing, pages 153–181. Kluwer Academic Publishing, 1996.

[35] Charles E. Perkins and Pravin Bhagwat, "Routing over multi-hop wireless network of mobile computers", In Tomasz Imielinski and Henry F. Korth, editors, Mobile Computing, pages 183–205. Kluwer Academic Publishing, 1996.

[36] C-K Toh,"The Cambridge Ad Hoc Mobile Routing Protocol", in Wireless ATM and Ad Hoc Networks, Kluwer Academic Publishers, Boston, Chapter 9, (1997).