Trends in Middleware for Mobile Ad Hoc Networks

Salem Hadim, Jameela Al-Jaroodi and Nader Mohamed
Department of Electrical and Computer Engineering
Stevens Institute of Technology, Hoboken, NJ 07030, USA
{shadim, Jaljaroo, nmohamed}@stevens.edu

Abstract— **The use of middleware has extended from simply facilitating applications' communication to a broad set of services supporting a huge spectrum of networked and distributed computing environments. At the same time mobile wireless ad hoc networks (MANET) have become a popular distributed environment and its application domain is expanding rapidly. However, like all distributed environments several issues must be considered and many problems have to be addressed to have efficient and useful applications. Current researchers moved towards using middleware to provide solutions to these issues and simplify application development for MANETs. In this paper we cover the latest trends and developments in middleware for MANETs and survey these approaches to identify their qualities and limitations. We then classify these approaches into six categories that group them based on the communication models and the programming paradigm used. The paper also evaluates these approaches in terms of the identified categories such as support for mobility, openness and heterogeneity, and the ease of use. Finally, we try to identify the open issues and the possible research directions that would provide better middleware solutions for MANETs.**

Index Terms— **Middleware, Mobile Ad Hoc Networks, Pervasive Computing, Programming Paradigms**

I. Introduction

In our future living environments information will be the most important commodity and exchanging via electronic devises (wired and wireless) and across networks will be a very important factor. It is foreseeable that the interaction of computing and communication will increase drastically and devices (e.g. home appliances and computing devices embedded in cars) incorporating microprocessors with communications capabilities will be very common. This will extend the communication field to a fully pervasive computing environment relying heavily on wireless ad hoc networks. Unlike a fixed wireless network, wireless ad-hoc or on-the-fly networks (MANET) are characterized by the lack of infrastructure and centralized authority. Nodes in a MANET are free to move and organize themselves in an arbitrary fashion. MANETs are very dynamic in terms of available communication partners, network resources, connectivity and bandwidth. Furthermore, end-user devices are very heterogeneous, ranging from high-end laptops to low-end PDAs and mobile phones. In addition, the resources are limited in terms of available memory, CPU speed and battery power. Furthermore, the available bandwidth is much smaller compared to wired networks.

There is a broad spectrum of potentially useful applications for MANET, but application development in this domain is not easy. Obviously, solving the same issues in every new application from scratch is not feasible. Instead, middleware services that support development of applications for mobile ad-hoc networks is a novel approach that will offer information access and sharing, and considerable flexibility for MANET. Due to the unique characteristics of MANETs, traditional middleware solutions that assume a relatively fixed network infrastructure are not suitable. Most traditional paradigms adopt synchronous models of communication and generally are not resource aware. In recent years, interest has grown in designing a middleware layer that fully meets the needs of MANET applications. Here we investigate some representative middleware solutions for MANETs, evaluate and compare them. Furthermore, we propose a classification based on the programming approach used and the model of communication. This will allow the reader to get a clearer insight and basic understanding of the current and proven effective ways to tackle issues on the design of middleware for MANETs. The selection of the approaches is based on how innovative they are in supplying new concepts and solutions to cope with ad hoc scenarios requirements.

The remainder of the paper is structured as follows. Section II surveys some related work carried out in the field then section III outlines the most relevant challenges faced in middleware design for MANETs. In Section IV we describe the research projects and approaches. In Section V we propose a state of the art classification, then

Based on "Middleware Issues and Approaches for Mobile Ad Hoc Networks", a preliminary version published in the proceedings of the third IEEE International Conference on CCNC 2006, Las Vegas, Nevada, January 2006. Authored by Salem Hadim, Jameela Al-Jaroodi, and Nader Mohamed..

an evaluation framework is provided on which the projects are then evaluated and compared. In section VI we discuss some of the open issues, and then we conclude the paper in Section VII.

II. Discussion and related work

Here we discuss some existing work and aspects related to our research in middleware for mobile ad hoc networks. First it is important to mention that limited work has been done on surveying middleware for mobile distributed systems in general and MANETs in particular due to the complexity and the non obvious taxonomy of the available middleware solutions. The survey [13] constitutes a thorough study of middleware for distributed systems whether fixed, nomadic or mobile. However, to some extent, the paper failed to survey many relevant approaches with proven results in an ad hoc environment. Our work focuses only on pure ad hoc environments and provides a selection of well suited models and approaches in the field. Several middleware solutions have been proposed for distributed systems, generally with heavy computational load often adapting synchronous communication style. These approaches are more suited for fixed distributed systems since devices are resources-rich and high steady bandwidth is assured by the wired links. Examples of such approaches are: Object Oriented middleware such as CORBA [21], Microsoft COM [24] and Sun Java/RMI [19], Message Oriented Middleware such as IBM queries [22], Sun’s java message queue [15], and Transaction Oriented Middleware such as BEA”S Tuxedo [11]. In mobile distributed systems, the middleware should be computationally lightweight due to the scarce resources of the mobile devices and asynchronous since low bandwidth and disconnections are a norm. Another important issue is awareness of the dynamic context of the environment that should be provided using middleware techniques and application adaptation to come up with the optimal solution. An attempt of adapting traditional middleware solution on nomadic and mobile environments has been carried out such as IIOP [10] (Internet Inter ORB Protocol) which is a part of CORBA. Mobiware [2], Alice [10] and DOLMAN [23] also cover relevant work for nomadic or semi fixed distributed systems. Alternative and totally new approaches have been investigated for mobile distributed systems. The most relevant ones are: Odyssey [26], coda[27], bayou [28] Tspaces [29], salutation[25] and Jini [3]. Even though these approaches moved a step forward in adapting to a mobile environment, they failed to address pure ad hoc environments since most of them rely on a semi fixed architecture (i.e. cellular networks), and more resource-rich devices. In section IV we survey middleware solutions that have been demonstrated to better fit in a pure ad hoc scenario.

III Middleware Challenges for MANETS

The design and development of a successful middleware layer for MANETs is not trivial. It has to deal with many challenges dictated by the MANET characteristics on one hand and the applications requirements on the other:

Heterogeneity: The middleware should provide low level programming models to meet the major challenge of bridging the gap between hardware’s raw potential and the needed activities. It should establish system-level mechanisms interfacing to the various types of hardware and network systems, only supported by basic distributed primitive operating system abstractions. This will support a wide range of applications and hardware platforms.

Mobility and Network Topology: Due to the dynamic nature of a MANET, it exhibits frequent and unpredictable topology changes. The mobile nodes dynamically establish routes among themselves as they move; moreover a user in a MANET may not only operate within the ad-hoc network, but may also require access to a public fixed network. MANETs therefore, should be able to adapt the traffic and propagation conditions to the nodes’ mobility patterns.

Scalability: If an application gets bigger, the network should be flexible enough to allow the addition of more nodes anywhere any time without affecting the network performance. Efficient middleware services must be capable of maintaining acceptable levels of performance, as the network grows larger.

Limited Resources: Middleware should provide mechanisms for efficient use of processing, memory and communication resources, while maintaining low power consumption. A node should accomplish its basic operations [20] without resources exhaustion. As an example of energy aware middleware, most of the device’s components including the transceivers should be automatically turned on and off based on the application requirements.

Quality of Service: An important and unique property of middleware for MANETS is dictated by the design principles of application knowledge [13]. However middleware has to include mechanisms for injecting application knowledge in the infrastructure of the network. This allows mapping application communication requirements to network parameters for fine-tuning the network monitoring process. Most ad hoc network applications dictate minimum quality of service (QoS) requirements sustained over an extended period of time. Middleware should be able to support QoS and dynamically adjust to changes in QoS requirements.

Context awareness: It is a general term used to encapsulate almost all characteristics of mobile ad hoc applications. Context means every aspect that can impact the behavior of an application; therefore the middleware should be context aware [14]. We can distinguish two types of awareness: device awareness and environment awareness. Device awareness relates to the internal resources of the device: battery power, processing power, and memory. Environment awareness relates to external resources around the device such as network connectivity, bandwidth, location, and other hosts in range.

SECURITY: Providing communication among hosts in a hostile environment is a primary concern. Unique characteristics of MANETs pose various challenges to the security design such as open peer-to-peer (P2P) network architecture, a shared wireless medium and a highly dynamic topology. These challenges raised the requirement of developing secure solutions that achieve wider protection, while maintaining desirable network performance. There is no standard security mechanism in a MANET from the security design perspective to address this issue.

IV MIDDLEWARE APPROACHES FOR MANETS

In this section some design principles and research projects that have already been proposed will be presented. We will argue that middleware for MANETs are tightly coupled with applications and there is no single general middleware that resolves all problems. The surveyed middleware approaches are purely and specifically designed for an ad hoc environment. We exclude those designed for fixed, nomadic or semi ad hoc infrastructures which is beyond the scope of this paper.

A. STEAM [16]: It introduces the concept of Event Based Middleware in a mobile Ad hoc environment. It addresses some specific constraints related to MANETs. One of the constraints is that some middleware components of the event services cannot be located on independent physical machines. In addition, such components may not be co-located with mobile entities and pose problems regarding availability, consistency, coverage and computational resources. STEAM utilizes the implicit publish/subscribe model thus does not require separate dedicated fixed cluster of event servers for to operate as the case in P2P and mediator-based models. Significantly the implicit publish subscribe model allows the consumers to subscribe to particular event types and the publishing entities to publish events. The entities are therefore fully anonymous. STEAM uses the proximity (geographical and functional) group communication model. The geographical aspects specify the area where the information is valid and the mobile devices within the propagation range. The functional aspects represent the common interest of produces and consumers based on the type of information propagated among them. Using these two aspects the mobile devices discover each other and therefore communicate. STEAM supports three different types of event filters: Subject filters, Proximity filters and Content filters. The usage of content filters enables subscribing entities to express sophisticated queries, which enable fine grain filtering of events. The subject and proximity filters are utilized to address the scalability of the system. In STEAM subject and proximity filters are applied on the publisher's side. Events are only routed to subscribers if both filters match. On the other hand content filters are deployed at the subscriber's side and utilized when an instance of an event is received to determine whether or not to deliver the event to the application.

B. EMMA [18]: As discussed in earlier sections, one solution for designing middleware for MANETs is to find a way to adapt a well known middleware technique used in traditional systems. This has a clear advantage in allowing application developers to adopt the same standards on mobile and dynamic devices. Also it allows interoperability between the wired and the ad hoc infrastructures. Therefore using paradigms based on asynchronous mechanisms constitute an adequate solution for an ad hoc environment where frequent disconnection and bandwidth fluctuation are the norm. Message Oriented Middleware (MOM) is a popular paradigm. EMMA is a MOM based middleware that exploits the Java Message Service (JMS) originally deigned for semi mobile distributed systems. EMMA is an attempt to adapt the JMS to fit MANETs by providing a slight modification of the message passing used in JMS and adding an epidemic routing mechanism [18] that facilitate delivery of messages in a MANET environment. As in JMS, EMMA applications can use the point to point or the publish-subscribe communications style. In point to point, applications use queues for asynchronous message exchange between the producer and possible consumers. The optimal location of the queues is determined by a negotiation process that is application dependant, which makes the middleware context aware. To allow the hosts that are not within range to receive messages, the asynchronous epidemic routing protocol is used. Each host maintains a buffer of messages created and messages received and messages are dropped if the buffer overflows. As a result the reliability of this protocol a best effort one and it does not grantee that all messages are delivered. In the publish-subscribe model some hosts contain topics and subscriptions of hosts interested in the topic. Topics are exchanged through subscribed group members using a synchronous protocol or an epidemic protocol. This model also provides mechanisms for maintaining subscription and unsubscription messages.

C. Expeerience [4] among the key objectives of an ad hoc environment is information sharing, access and communication. On one hand this makes P2P communication an adequate approach to tackle many issues since it shares some key characteristics with MANETs. On the other hand they also have some key differences such as the dynamic topology and the lack of guaranteed connectivity. In addition, scalability in P2P is limited in terms of data rates, whereas in MANETs it is conditioned by low bandwidth and low processing power.

From this stems the design of Expeerience. It takes advantage of services provided by a P2P environment and adds various modules needed for MANET. The P2P framework chosen is JXTA [4] which offers key advantages such as interoperability, platform independence and ubiquity. JXTA tries to create a common platform for developing distributed P2P services and applications. Expeerience enhances some of the services in JXTA and adds another software layer that meets the requirement of MANET that is not met by JXTA. Thus introducing new features like: management of the intermittent connections and multiple interfaces, more efficient resource discovery mechanisms and code mobility. It is important to mention another concept used by Expeerience which is code mobility. This new paradigm allows the download and installation of new services dynamically. This feature allows the middleware to dynamically adapt to situations that were not considered during the design and that only take place at run-time.

D. SELMA[19]: A middleware platform that uses mobile agents communicating through the "marketplaces" pattern, in which mobile applications send and forward agents to specific geographic locations called marketplaces. In these well-defined locations, the probability of finding the required information is very high since they are characterized by a high presence of mobile hosts. This communication pattern is suitable for a large number of ad hoc applications such as online auctions and electronic billboards. SELMA middleware fits well in an ad hoc scenario by being self-configurable and power aware. It supports hop-to-hop communication and multi-hop communication as well. The user specifies the application agent in a mobile host and then the agent moves toward a specified target marketplace using the agent transport protocol to provide any service to the mobile host originator. After finishing its task in the marketplace, the mobile agent moves either to another marketplace to do another task or moves back to its originator using homezones. A homezone is defined as a geographical area with a high sejourn probability of the originator. The proposed middleware architecture is composed of three parts, communication abstraction, agent platform and the set of application and service agents. The communication abstraction layer provides different services such as mobile hosts positioning, wireless communication and neighbor discovery. The wireless communication model uses both local unicast and local broadcast to all one hop neighbors. The current implementation of SELMA is based on the ad-hoc mode of the IEEE 802.11. The agent platform layer provides mechanisms for reliable agent transport using agent duplication, map computation by dividing the area surrounding the mobile device into small rectangles, hot spot detection allowing the device to detect where the marketplaces are, geographic routing, marketplace communication enabling agents to communicate between each other in a specific marketplace, and localization services. Finally, the upper layer contains application agents for user-defined application, and service agents that are not assigned to any user and used to provide location based services such as duplicate elimination and load balancing.

E. Mobile Gaia [32]: To address pervasive and ad hoc computing environments, Mobile Gaia middleware adopts a component based approach. That is, application services are decomposed into smaller components that can run on a cluster of different heterogonous devices. This yields to considerable memory and power savings, since the middleware allows only the required component to be loaded and unloaded to a device depending on its role. The mobile ad hoc network is divided into active spaces or clusters, where each personal active space has a device coordinator or a cluster head, and a set of client devices. Mobile Gaia divides services into two main categories, when the services are provided by the cluster coordinator; the services are referred to as coordinator services, and when the services are provided by the client device, the services are referred to as client services. It is important to mention, that in a coordinator role, the device has additional responsibilities such as managing all services in the devices belonging to its cluster, integrating data collected from the client devices and maintaining their locations. As a communication model, Mobile Gaia adopts the traditional event based model, namely, publish-subscribe. When an application wants to send an event, it creates an event channel in the host device, and then the device notifies its cluster coordinator with the nature event using the event service. The cluster coordinator maintains a repository of events, and when an application is interested in any event, it subscribes to it by notifying the local event service which in its turn queries the event service in the cluster coordinator for that specific event. Mobile Gaia architecture is comprises a main Kernel and an application framework. The kernel contains a set of services and a service deployment framework responsible for installing new services, loading and unloading services when they are no longer needed. The frame is based on the "What You Need Is What You Get" model [34]. The application framework eases the development of the distributed application by supplementing a common interface offering context awareness and adaptation.

F. LIME [17]**:** is information sharing middleware. It adopts the tuple space based approach that stems from Linda [30], which provides tuple space data structure for fixed distributed systems. The tuple space systems have been demonstrated to provide many useful features for wireless environments. LIME extends the model adopted in Linda and makes it suitable for highly dynamic mobile environments. LIME defines a tuple space for each mobile host and permanently associates it with it. When a mobile host connects to other hosts, rules for transient sharing of the individual tuple spaces are defined. This is effective for a fast and accurate information exchange between mobile hosts when a connection is established. Each mobile host maintains an interface tuple (ITS) that is permanently and exclusively attached to that unit and transferred with it when movement occurs (like in the data tree of Xmiddle). Each ITS contains tuples that the unit wishes to share with others and it represents the only context accessible to the unit when it is alone. However, the content of the ITS is dynamically recomputed so it looks like the result of the merging of the ITSs of other mobile units currently connected. Upon the arrival of a new mobile unit, the content perceived by each mobile unit through its ITS is recomputed taking the content of the new mobile unit into account. This operation is called engagement of tuple spaces; the opposite operation, performed on departure of a mobile unit, is called disengagement. Information about the system configuration is made available through a read-only transiently shared tuple space called Lime System, containing details about the mobile components present in the community and their relationship. Moreover, reactions can be set on the tuple space, to enable actions to be taken in response to a change in the configuration of the system. An important aspect of Lime is tuple access and movement; events are used to notify users when a new tuple is available. It is worth noting that the tuple space approach is widely used by other middleware to support mobility such as TSpaces [29] of IBM and L2imbo [6].

G. LIMONE [8]: An enhancement of LIME middleware is investigated and implemented in Limone. One limitation of LIME is that it assumes that the network is not highly dynamic and can sustain a permanent connection during a group transaction. This symmetric approach enforced by group membership presents limitation in case of a highly mobile environment where permanent connections are not guaranteed. To cope with this issue, Limone accommodate a high degree of uncertainty in the state of a network by providing more robust interaction model between agents residing in hosts. The model is based on individual agents having full control on the distributed transaction it participates with. This is done by making each host maintain an acquaintance list that provides a global view of the operating context and is customizable using admission policies depending on the network dynamics and the application requirements. Limone's main features are context management, explicit data access, reactive programming and code mobility or agent migration. In respect to context management, an efficient discovery mechanism is provided to allow the agents to discover neighbors and to selectively decide on their relevance depending on the application requirements and network settings. This approach copes better with scalability, limited hardware resources and security issues.

H. MESHMdl [33]: Context awareness and self organization are the main design principles of the MESHMdl middleware to cope with mobile ad hoc scenarios. It is based on two well known approaches: autonomous mobile agents for *logical mobility* and tuple spaces [36] allowing decoupling applications components in time and space which is necessary to cope with the high dynamics of an ad hoc environment such as mobility and frequent disconnections. This asynchronous mode of communication allows different peers and mobile nodes to communicate without having to set up a "rendezvous". Applications in MESH*M*dl are expressed as groups of mobile agents that collaborate via spaces. The MESHMdl architecture is comprised of: a generic connection layer, an interaction layer, Space layer, agent runtime or Engine, and agent applications. The generic connection layer encapsulates different network technologies and supplements the upper layers with neighbor discovery mechanisms and a way to interact with them. This layer makes the middleware independent from the underlying network. The role of the space layer is to define the way spaces are managed and defined. It uses an object oriented implementation of the tuple space paradigm as adopted in JavaSpaces [37]. The space is considered as a shared communication medium whose role is twofold: inter-agent communication and agent Engine communication. The agent runtime or Engine serves as a runtime environment for agents and runs on every node hosting an agent. The agent applications are responsible for managing the creation of agent and the way they are cooperating to form an application. MESHMdl is mobility aware since spaces are not federated when two nodes meet as in LIME [18]. Instead they appear separate to the applications. An engagement protocol is used for communication when hosts are within communicating range. Furthermore, and information diffusion is supported, namely, the Xector model which is a mechanism used to propagate information through the network under certain constraints.

I. XMIDDLE [14]*:* Considered as data-sharing-oriented middleware, it provides mechanisms to deal with frequent disconnected operations in an ad hoc environment. Xmiddle moves a step forward by providing a robust data structure to deal with ad hoc scenarios and no assumption is made about the existence of fixed provider or privileged nodes as in the case in nomadic ones. Using a tree like data structure, mobile hosts can establish efficient communication with each other. Each mobile host maintains a private tree data structure with access points to allow communication with other hosts. XMIDDLE therefore provides an approach to sharing that allows on-line collaboration, off-line data manipulation, synchronization and application dependent data reconciliation. As long as two hosts are connected, they can share and modify the information on each other's linked data trees. When disconnections occur, the disconnected hosts retain replicas of the trees they were sharing while connected, and continue to be able to access and modify the data. When the two hosts reconnect, the reconciliation is accomplished using the replicas of the tree previously stored. This allows restoring only a small specific part of the tree rather than the whole tree. This turns to be a critical issue since not all devices has sufficient resources to update the whole tree at all times. The tree data structure is implemented using XML [14] and related technologies. Data is stored in XML documents, which can be semantically associated to trees. Nodes, address branches and references in the XML documents are then manipulated using technologies such as Document Object model, Xpath and Xlink. It is worth noting that representing mobile data structures in XML enables seamless integration of Xmiddle applications with the various systems and Micro Browsers.

J. Mate [12]**:** Mate is among the middleware for wireless sensor networks (WSN) that uses a virtual machine (VM) approach as an abstraction layer to implement its operations and to tackle the different challenges of WSN, which carry great similarities to MANETs. Mate focuses on the need for new programming paradigms to overcome constraints such as limited bandwidth and the large energy draw from network activity. Mate proposes a spectrum of reprogrammability from simple parameters adjustments to uploading complete program updates using a VM approach. The energy cost of sending a single bit of data can consume the same energy used to execute thousands of instructions. A content specific routing and reprogramming model can be used and supported by the VM. Mate is a byte code interpreter built on TinyOS [12] operating system designed specifically for sensor networks that run on motes (small devices with a small CPU and limited storage resources) to implement the middleware operations. It uses codes that are broken into capsules of 24 instructions, each of which is a single byte long. This gives the advantage to large programs to be decomposed to multiple capsules, thus easy to inject into the network. The key components are the VM (Mate), Network, Logger, Hardware and Boot/Scheduler. Using a synchronous model that begins execution in response to an event such as packet transmission or timer signal, it avoids message buffering and large storage. The synchronous model makes application-level programming simpler and less prone to bugs than asynchronous event notifications. Another key functionality of Mate is infection or network updates done by using version numbers, so comparison could be made at the neighbors and the newest version is installed.

V. Classification , Evaluation and Comparison

In the previous section we surveyed different existing middleware approaches, based on the programming models used. In this section, we will classify the available approaches. Then we will define an evaluation framework, which we will use to evaluate the different systems under consideration.

A. *Classification*

Based on the information gathered we may classify the approaches into six main categories: Event based and Message Oriented Middleware (MOM), component based and Mobile agents middleware, Peer to peer based middleware, tuple spaces based middleware, data sharing based middleware, and virtual machine based middleware. It is important to mention that while all categories differ in their objectives and the way they deal with programming a mobile ad hoc network as whole; they may have some common features, which make them seem to be non mutually exclusive. This is because research is still maturing and some desired features are sometimes common between some categories.

- *Event Based and Message Oriented Middleware (MOM):* Used in distributed systems exhibiting *Event*-based architectural style. Event based systems are particularly adequate for distributed environments without central control such as mobile ad hoc networks to support applications that must monitor or react to changes in the environment and information interest. One sub area of interest is MOM, which is a communication model in a distributed mobile ad hoc network. It facilitates message exchange between mobile nodes using the publish-subscribe mechanism. The strength of this paradigm lies in that it supports asynchronous communication very naturally allowing loose coupling between the sender and the receiver. This turns to be very suitable in pervasive environments such as MANETs

- *Peer-to-Peer (P2P) Middleware*: P2P is an interaction model between end points and allow them to share information or accomplish a common task. In P2P architecture, no single host is permanently seen as a server, but each single host is able to play both the role of server and client according to the user's and application's needs. This decentralized architecture makes it a suitable backbone for ad hoc environments. Hence appropriate middleware is needed to provide abstractions to the upper applicative layers and cope with high dynamics such as mobility and resource discovery.

- *Component-based and Mobile Agents Middleware*: The key in this approach is that applications are as modular as possible then injected and distributed through the network using *mobile agents'* migration or diffusion protocols. This yields to considerable energy savings since transmitting small modules will require less energy consumption than a whole application. For example, Mobile Gaia provides a component architecture where components can be installed or loaded/unloaded to dynamic applications. Its autonomic behavior increases its fault tolerance and self-organization of the network.

- *Tuple Spaces Based Middleware:* as mentioned earlier, ad hoc environments are characterized by low and variable bandwidth, frequent disconnections…etc. Thus a decoupled and opportunistic style of communication is required. Decoupled means that communication happens even in the presence of disconnections, and opportunistic as it exploits connectivity whenever available. One solution offering this possibilities is Tuple Spaces introduced in the Linda[30] coordination language.

• *Data Sharing Based Middleware:* This approach deals with frequent disconnections and data sharing by providing robust mechanisms and data structures to *maximize the availability* of the data, providing the users with data replicas. The main goal is to tackle issues such as detecting and resolving conflicts that occur in ad hoc systems by ensuring the integrity and consistency of those replicas. The drawback of this approach is that it is only suitable for small scale networks since only a subset of hosts are able to access a small amount of networked data.

• *Virtual Machine Based Middleware*: It is a flexible system containing virtual machines (VMs), interpreters and mobile agents. It allows applications to be written in separate small modules that get injected and distributed through the network using tailored algorithms such as overall energy consumption and resource use are minimized. The modules then get interpreted by the virtual machine. The approach however, suffers form the overhead introduced by the instructions.

B. Evaluation

Before evaluating each middleware, we first define a framework for the evaluation based on the following criteria: heterogeneity, scalability, power awareness, Mobility, openness and ease of use. The first four criteria are already defined in the third section; likewise we define openness and ease of use, which are very important in the notion of middleware.

EASE OF USE*:* Refers to the level of abstraction of the middleware. In other terms, how its interface relieves the user from dealing with the complex low level APIs of heterogeneous resources by providing an easy to use "single entity" of the whole system. Large-scale ad hoc network applications usually involve various resource types; this increases the complexity of the code and the programming implementation to manage all these heterogeneous resources. For example environment monitoring and multimedia involves a number of heterogeneous resources, including network bandwidth, CPU Cycles, memory buffers, and storage. Dealing with this heterogeneity will require *abstract* resource models to alleviate the complexity of coordinating and adapting these diverse resources. Such models should be uniform and easy to use offering a user-friendly interface.

OPENNESS*:* Is the possibility to extend and modify the system easily, as a consequence of changed functional requirements. The system should support both environment awareness and device awareness. Traditional approaches encapsulating and hiding the implementation details result in a "black box" that is difficult to inspect and modify. This should be avoided in middleware design for Manets where dynamic topology and frequent resource changes are a norm rather than an exception. For example in devices such as PDA and sensors, which have limited battery life, CPU power and memory capacity, the system resources should conform to the constraints dictated by the deployment platform and network topology. Designing an open middleware prevents stagnation. Resource management in middleware should *dynamically adapt* to resource availability and other contextual changes. Standards are essential for open systems and must be continually updated as the environment evolves. Developers must introduce open resource configuration and reconfiguration to achieve the resource management adaptation that the middleware requires.

Following, we evaluate each considered middleware by concentrating on how well they meet the criteria defined in the framework and focusing on the advantages and disadvantages of each approach.

STEAM provides an event service communication middleware suitable in an ad hoc scenario. However, its publish/subscribe mechanism is limited by proximity of nodes and that they must be within reach of each other. It does not provide efficient resource discovery mechanisms to support high nodes *mobility*. In STEAM each device must be able to perform complex content filtering and this may not be possible for *resources constrained* devices. It addresses *scalability* when the mobile hosts are within a proximity communication group; however *scalability* problems can arise when it comes to distributed applications using entities which are not in close proximity. This significantly limits the usefulness of STEAM in terms of application *heterogeneity*. Indeed, STEAM has been designed with the traffic management application in mind. *Openness* and context awareness of the middleware is limited since no mechanisms have been provided to handle failed and temporary unavailable entities. Finally, applications are expressed in terms of events, which is a well know paradigm and *easy to use.*

EXPEERIENCE is a middleware layer over JXTA that addresses issues with regard to intermittent connections in ad hoc environments. Expeerience moves a step forward by containing efficient features to cope with ad hoc scenarios. It supports code mobility and service migration, including support for mobile agent systems allowing *scalability*. The new enhanced resource discovery service component allows better disconnection management and the discovery of distributed agents hence the *mobility* is addressed. This also enables *power and hardware resources awareness*, such as the life time of mobile nodes is increased. The code mobility concept makes the middleware adaptive where new services could be added at run time, hence *openness* is supported. Expeerience does not, however, address the component models issue with JXTA nor protocol exchangeability. In its present state Expeerience uses some libraries written in J2SE and C, a light weight version using J2ME or lighter version of the JXTA middleware would be needed to support *heterogeneity* and to include devices such as PDAs. This layer is also needed to supplement the application developer with an *easy to use* single interface and programming primitives.

EMMA uses an adapted Java Message Service (JMS) armed with two communication styles: point-to-point and publish-subscribe to cope with ad hoc scenarios such as intermittent connectivity and partial *mobility*. An epidemic routing mechanism is added to support message delivery to mobile hosts that are not in reach. However, the poorly performing epidemic algorithm in terms of the number of replicas that are spread across the network dictates that a tradeoff between application level routing and *resource* such as *power* and bandwidth usage should be investigated. This also poses *scalability* and *openness* issues for dynamic network adaptation since the middleware doesn't provide other resource discovery mechanisms beside the epidemic routing protocol. Another limitation is that the reliability offered is a best effort one, which results in the loss of some messages. A prototype of EMMA has been implemented using J2ME - a virtual machine implementation-, which is suitable for *heterogeneous* devices such as PDAs and laptops. The combination of the MOM paradigm and J2ME offer high level abstractions and an easy to use interface.

SELMA: Using mobile agents under the marketplace and homzones patterns, SELMA addresses well *mobility*. Self organization, *scalability* and adaptation are the main design principles of the middleware. This is done by incorporating a set of services such as neighbor discovery, map computation, load balancing and geographic routing. The communication abstraction provides generic methods for positioning, wireless communication and device discovery, thus the middleware is portable across different communication hardware and addresses *heterogeneity*. The neighbor discovery service allows considerable power and *resource savings* by collecting information on the devices capacities in hosting additional agents and communicating it to the middleware. The programming abstraction used by SELMA is quite *easy to use*, however one limitation of the middleware is that dynamic creation of new marketplaces is till under investigation, thus its *openness* is partial. Furthermore, SELMA was evaluated only on application following the marketplace pattern making the middleware not general purpose.

MOBILE-GAIA combines three main design pillars, decomposing application into components to run on different device, thus saving some *power* and hardware resources, event based service, namely publish-subscribe very suitable for *mobility* supported with the discovery and cluster management service, active spaces supporting *scalability* with novel coordinator-client roles. The service deployment framework allows news services to be installed, components to be loaded and unloaded depending on application needs making the middleware context aware and *open*. Muti-device support and environment independence is addressed in Mobile Gaia by offering common programming patterns and by using the "WYNIWYG" model, thus supporting *heterogeneity* and making it *easy to use*. However, it is important to mention that some mechanisms are under investigation to allow locating services depending on devices memory capacity, bandwidth and power.

LIME introduces a new approach in designing middleware services for ad hoc environments which is data sharing based on tuples spaces. However, LIME mobile hosts are connected only when the distance between them allows direct communication using events notification. Mobile agents are connected when they are co-located on the same host, or they reside on hosts that are connected. This turns to be a serious limitation for ad hoc applications where efficient multi hop communication mechanisms should be provided to support high mobility. That is, in its present state LIME is only supports partial physical mobility between host and logical mobility between agents. Furthermore, Lime provides some context awareness but the overhead cost is very high; the blocking behavior of its primitives adds to the overhead. There is no support for behavior *adaptation* .It doesn't constitute a full middleware package designed to meet all MANETs requirements.

MESHMdl is built on two models, mobile agents and tuple spaces. To adapt to physical *mobility*, MESHMdl uses logical mobility by agent migrations. Applications components are decoupled in time and space, therefore an asynchronous model of communication is used which is suitable for saving some power and network resources such as bandwidth. So application can migrate logically in case of failure or mobile device going out of range. This confers to the middleware some openness and context awareness. Unlike Lime the tuple spaces are *scalable* since the node-level spaces are not federated when two nodes meet. The tuple spaces are object oriented therefore *easy to use*. The middleware supports heterogeneity is compliant with java2 Micro Edition specification and the Connected Limited Device Configuration from Sun [35], and can run on PDAs , laptops and desktops.

XMIDDLE provides an enhancement for information sharing mechanisms between mobile hosts. Using an enhanced data structure based on trees implemented in XML and other technologies. It moves a step forward in addressing mobile computing issues such as scarce device *resources* and frequent disconnection. Partial *mobility* is supported using a tree like data structure allowing better information sharing between connected hosts. Frequent disconnections are handled using different protocols such as link and reconciliation protocols. The replication protocol enables keeping a copy of the communication data structure when disconnections occur so that no updates are needed when the connection is reestablished. This enables considerable *energy savings* and fast synchronization. XMIDDLE is implemented in Java and relies on the virtual machine, which makes it platform independent and easily supports *heterogeneity*. However, XMIDDLE suffers some limitations that require further investigation: the communication paradigm (i.e., sharing of trees) is basic and needs to be enhanced to model more complex ad hoc mobile interactions making its *openness* very limited. Also a key limitation of XMIDDLE like LIME is that multi-hop scenarios are not considered where routing through mobile nodes is required. Also resource discovery mechanisms should be provided to meet requirements such as *scalability*. Finally, XMIDDLE uses very *easy to use* programming abstraction and XML and java are considered very high level languages.

MATE with its virtual machine approach supports *scalability* and *openness* by the use of active messages to update the network protocols and parameters by injecting new capsules. This makes the network dynamic, flexible and easily reconfigurable.. Mate gives a user–land supplemented by the VM, hence supports *heterogeneity* and provides efficient network and sensor access. *Mobility* is addressed by using various ad hoc routing protocols and protocol updates. However, In terms of energy *-power awareness-*, Mate is only suitable for sleepy applications that are in low duty cycle most of the time, for complex applications, it is wasteful because of the interpretation overhead of its instructions. Also in its current state, Mate is only architecture and byte codes; making it not *easy to use*; a higher-level language and programming model for application development are needed.

Table 1 summarizes the evaluation of the different approaches based on the proposed framework. (F indicates full support; P, partial support; and X, little or no support)

Table. 1. Approaches of Mobile Ad Hoc Networks Middleware.

Project Name	Main Features	Power Awareness	Openness	Scalability	Mobility	Heterogeneity	Ease of use
Event Based and Message Oriented Middleware							
STEAM[16]	Event based, Proximity Group Communication, Filters, Publish-Subscribe Mechanism	P	X	P	P	P	F
EMMA[18]	Message Oriented Middleware, JMS, Point to Point Communication, Publish, Epidemic Routing	P	X	P	P	F	F
Peer to Peer Based Middleware							
Expeerience [4]	JXTA, P2P Framework, Mobile Code, Resource Discovery Mechanisms, management services	P	F	F	F	P	F
Component and Mobile Agents Based Middleware							
SELMA [31]	Marketplace pattern, Mobile Agents, Homzones, Neighbor discovery, IEEE 802.11, duplication.	P	P	F	F	F	F
Mobile-Gaia [32]	Small Components, Active Spaces, Clusters, Publish-Subscribe, "WYNIWYG", Coordination	P	F	F	F	F	F
Tuple Space based Middleware							
LIME [17]	Data Sharing Middleware, Shared Tuple Spaces System, Extends Linda, Interface Tuple (ITS)	P	X	P	P	P	F
MESHMdl[33]	Object Orineted Tuple Spaces, Mobile Agents, J2ME, Asynchronous, Xector model.	P	P	F	F	F	F
Information sharing Based Middleware							
XMIDDLE[14]	Data Sharing Middleware, Robust Tree-like Data Structure, XML, Management of Disconnections	F	X	P	P	F	F
Virtual Machine Based Middleware							
Mate [12]	Uses TinyOS, Synchronous, Byte code interpreter, Mobile active capsules.	F	F	F	P	P	X

VI. Discussion and Open Issues

The middleware approaches and projects surveyed in this paper all provide different mechanisms and techniques to tackle different challenges and impediments of the design and development of a middleware for MANETs. However, a close examination based on our evaluation in the previous section reveals that the approaches are tightly coupled with specific applications and none fully meets the challenges presented in section IV, more specifically context awareness, QoS, heterogeneity and efficient resource discovery.

We believe that a complete and effective middleware should combine more than one approach and mechanism to cover a wide range of ad hoc requirements. Middleware implementation using virtual machines and using mobile code techniques adopting an asynchronous model of interaction between hosts such as the message oriented style would offer many potential solutions and drastically enhance middleware possibilities in ad hoc environments. In addition, keeping the middleware lightweight and taking the context awareness as a functional requirement throughout the design and development of the middleware is also essential for success. Indeed, mobile code techniques allow creating new services and migrating services at run time and deal with issues that could not be predicted in the design phase. Furthermore, efficient adaptable resource discovery mechanisms specifically adequate for a mobile ad hoc environment should be provided to give the middleware more robustness and flexibility in handling the frequent changes in the network components and the topology.

Security requirements pose another major challenge to address in MANETs. The security mechanisms used and proven in traditional networks are not suitable for Manet's [1]. The absence of centralized authority, dynamic network topology and mobility cause serious problems. In addition, the limited resources and device independence dictate the need for new sophisticated solutions that should be incorporated in the design of middleware for MANETs. These techniques must be capable of functioning efficiently on the independent devices, while keeping resource consumption as low as possible.

VII. CONCLUSION

In this paper, we surveyed different middleware approaches specifically adopted for wireless mobile ad hoc networks. We studies some of the issues involved and tried to clarify some of the ambiguities of middleware definitions. Then we identified the major challenges that the design and development of middleware for MANETs faces. Furthermore, we investigated many of the relevant existing projects carried out towards this perspective. We provided a thorough evaluation and comparison by concentrating on similarities and differences between the approaches. In addition, we tried to provide an overview of the positive features and advantages along with the shortcomings and disadvantages of the approaches studied. We were able to identify the following distinct approaches: event based and message oriented (MOM), P2P, component and mobile agents based, tuple spaces and information sharing. Some of the approaches are based on a virtual machine and mobile code techniques. Furthermore, and based on the results of our comparison, we discussed and proposed potential enhancements and new research possibilities in the field. At the end it is important to mention that designing and implementing the middleware that fully meets all the requirements and challenges of a mobile ad hoc environment is to some extent not a realistic venture. Trade-offs must be made to reach a more realistic approach that incorporates various techniques and methodologies to provide as many of the required functionalities as possible, while maintaining flexibility, efficiency, and scalability.

ACKNOWLEDGMENT

The authors wish to thank the editors of the Journal of Communications (JCM) for their invitation to contribute with this paper.

REFERENCES

[1] J. Al-Jaroodi, "Security issues at the network layer in wireless mobile ad hoc networks", in proc. international conference on wireless networks (ICWN'05), Las Vegas, Nevada, June 2005.

[2] O. Angin, A. Campbell, M. Kounavis and R. Liao. "The Mobiware Toolkit: Programmable Support for Adaptive Mobile Netwoking," in Personal Communications Magazine, SI on Adapting to Network and Client Variability. IEEE Computer Society Press, August 1998.

[3] K. Arnold, B. O'Sullivan, R. W. Scheifler, J. Waldo, and A. Wollrath., "The Jini [tm] Specification.," Addison-Wesley, 1999.

[4] M. Bisignano, A. Calvagna, G. D. Modica, O. Tomarchio, "Expeerience: a Jxta middleware for mobile ad hoc networks," in proc. third international conference on P2P computing, 2003.

[5] L. Capra, C. Mascolo, S. Zachariadis and W. Emmerich., "Towards a Mobile Computing Middleware: A Synergy of Reflection and Mobile Code Technique," In Proc. 8th IEEE Workshop on Future Trends in Distributed Computing Systems, Italy 2001. pp. 148-154.

[6] N. Davies, A. Friday, S. Wade, and G. Blair," L2imbo: A Distributed Systems Platform for Mobile Computing," ACM Mobile Networks and Applications (MONET), Special Issue on Protocols and Software Paradigms of Mobile Networks, 1998, 3(2).

[7] W. Emmerich. "Software Engineering and Middleware: A Roadmap", In the Future of Software Engineering. A. Finkelstein (ed), ACM Press, 2000, pp: 117-129.

[8] C.-L Fok, G.-C. Roman, G. Hackmann, "A Lightweight Coordination Middleware for Mobile Computing," in proc. 6th Intern'l Conf. on Coordination Models and Language, De Nicola, R., Ferrari, G., and Meredith, (editors), Lecture Notes in Computer Science 2949, Springer-Verlag, Pisa, Italy, Feb. 2004, pp. 135-151.

[9] D. Gelernter, " Generative Communication in Linda", ACM Trans. On Programming Languages and Systems, 1985,7(1), pp. 80–112.

[10] M. Haahr, R. Cunningham, V. Cahill."Supporting CORBA Applications in a Mobile Environment (ALICE)" in proc. 5th Int. Conf. on Mobile Computing and Networking (MobiCom). Aug. 1999.

[11] C. Hall, "Building Client/Server Applications Using TUXEDO." Wiley, 1996.

[12] P. Levis, D. Culler, "Mate: A Tiny Virtual Machine for Sensor Networks," in proc. Inter. Conf. on Architectural Support for Programming Languages and Operating Systems, Oct. 2002.

[13] C. Mascolo, L. Capra, W. Emmerich, "Middleware for Mobile Computing", Adv. Lectures on Networking, E. Gregori, G. Anastasi and S. Basagni (ed.), Lecture Notes in Computer Science, Springer Verlag, 2002. vol 2497, pp. 20-58.

[14] C. Mascolo, L. Capra, S. Zachariadis, W. Emmerich, "XMIDDLE: A Data-Sharing Middleware for Mobile Computing," in Wireless Personal Com., Kluwer. 2002. 21. pp. 77-103.

[15] R Monson-Haefel, D. A. Chappell, M. Loukides, “Java Message Service », O’Reilly & Associates, Dec. 2000.
[16] R. Meier, V. Cahill, “STEAM: Event-based middleware for wireless ad hoc networks," in the 22nd Intern. Conf. On Distributed Computing Systems Workshops (ICDCSW '02), Austria, July 2002.
[17] A. L. Murphy, G. P. Picco, G.-C. Roman. “Lime: A Middleware for Physical and Logical Mobility,” in proc. of the 21st Intern. Conf. On Distributed Computing Systems (ICDCS-21), May 2001.
[18] M. Musolesi, C. Mascolo and S. Hailes, “EMMA: Epidemic Messaging Middleware for Ad Hoc Networks,” in Personal and Ubiquitous Computing. 2005.
[19] E Pitt, K. McNiff, “Java rmi: The Remote Method Invocation Guide,” Addison Wesley, June 2001.
[20] T Plagemann, ET. al., “Towards Middleware Services for Mobile Ad-Hoc Network Applications,” in proc. of the IEEE Workshop on Future Trends of Distributed Computing Systems (FTDCS’03), 2003.
[21] A Pope., “The Corba Ref. Guide: Understanding the Common Object Request Broker Architecture”, Addison-Wesley, Jan. 1998.
[22] I. Redbooks, “MQSeries Version 5.1 Administration and Programming Examples, “IBM Corporation, 1999.
[23] P Reynolds and R. Brangeon, “Service Machine Development for an Open Longterm Mobile and Fixed Network Environment”, http://www.fub.it/dolmen/, 1996.
[24] D. Rogerson. “Inside COM,” Microsoft Press, 1997
[25] Salutation Consortium. , http://www.salutation.org/,96
[26] M. Satyanarayanan., “Mobile Information Access”, in IEEE Personal Communications, Feb. 1996. 3(1):26–33.
[27] M. Satyanarayanan, J. Kistler, P. Kumar, M. Okasaki, E. Siegel, D. Steere, “Coda: A Highly Available File System for a Distributed Workstation Environment”, in IEEE Trans. on Computers, Apr. 1990, 39(4), pp. 447–459.
[28] D. Terry, M. Theimer, K. Petersen, A. Demers, M. Spreitzer, C. Hauser, “Managing Update Conflicts in Bayou, a Weakly Connected Replicated Storage System.”, In proc. of the 15th ACM Symp. On Operating Systems Principles, Colorado, Aug. 1995, pp. 172–183.
[29] P. Wyckoff, S. W. McLaughry, T. J. Lehman, D. A. Ford, “T Spaces,” IBM Systems Journal, 1998, pp. 454–474
[30] D.Gelertner, “Generative Communication in Linda”, ACM Computing Surveys, Vol 7, No 1, pp.80-112, Jan. 1985.
[31] Daniel G¨orgen and Hannes Frey and Johannes K. Lehnert and Peter Sturm. SELMA: A Middleware Platform for Self-Organizing Distributed Applications in Mobile Multihop Ad-hoc Networks. In Western Simulation MultiConference WMC ’04, 2004.
[32] Chetan Shankar, Jalal Al-Muhtadi, Roy Campbell and M.Dennis Mickunas, "Mobile Gaia: A Middleware for Ad-hoc Pervasive Computing", IEEE Consumer Communications & Networking Conference (CCNC 2005), Las Vegas, Jan. 2005.
[33] K. Herrmann, “MESHMdl - A Middleware for Self-Organization in Ad hoc Networks," in Proceedings of the 1st International Workshop on Mobile Distributed Computing (MDC'03), May 19 2003.
[34] F. Kon, T. Yamane, C. Hess, R. Campbell, and M. D.Mickunas, "Dynamic Resource Management and Automatic Configuration of Distributed Component Systems," presented at 6th USENIX Conference on Object-Oriented Technologies and Systems (COOTS'2001), San Antonio, Texas, 2001.
[35] Sun Microsystems, “CLDC - The Inner Plumbing of the Java 2 Platform, Micro Edition.” http://java.sun.com/products/cldc (official website).
[36] S. Ahuja, N. Carriero, and D. Gelertner, “Linda and friends,” IEEE Computer, pp. 26–32, August 1986.
[37] Sun Microsystems, “JavaSpaces[tm] Service Specification, v1.2.1.”www.sun.com/software/jini/specs/js1_2_1.pdf, April 2002.

Salem Hadim is currently pursuing his PhD degree at The Department of Electrical and Computer Engineering, Stevens Institute of Technology, New Jersey, USA. His main research interests include middleware, software systems, Security, Networking, Autonomic Computing and Enabling Technologies. He received the MS degree in Computer Engineering from Stevens Institute of Technology, New Jersey, USA, in 2003. As student member of IEEE and IEEE Communication Society, he has authored/co-authored several papers in international Journals and conferences.

Jameela Al-Jaroodi is a Research Assistant Professor at The Department of Electrical and Computer Engineering, Stevens Institute of Technology. Her research interest is in distributed systems including middleware, distributed software for heterogeneous systems, ad-hoc networks, sensor networks, embedded software, and software engineering for distributed systems. She published more than 30 articles in Journals and Conferences. Dr. Al-Jaroodi received her doctoral degree from The Computer Science and Engineering Department at The University of Nebraska-Lincoln in 2004; her master degree from Western Michigan University in 1998; and her bachelor degree from The University of Bahrain in 1993.

Nader Mohamed is an Assistant Professor of Electrical and Computer Engineering, Stevens Institute of Technology, New Jersey, USA. His main research interests include middleware, networking, real-time and embedded software, cluster and Grid computing, and dependable systems. He obtained a Ph.D. in Computer Science from University of Nebraska-Lincoln; 2004, an M.Sc. in Computer Science from Western Michigan University; 1998, and a B.Sc in Electrical Engineering from University of Bahrain; 1992. He has also over eight years of industry experience in systems integration and middleware.