IADIS International Journal on Computer Science and Information Systems
Vol. 2, No. 2, pp. 65-82
ISSN: 1646-3692

A CONTEXT AWARE INFORMATION SHARING MIDDLEWARE FOR A DYNAMIC PERVASIVE COMPUTING ENVIRONMENT

Addisalem Negash, Vasile-Marian Scuturici and Lionel Brunie *Laboratoire d'InfoRmatique en Images et Systèmes d'information (LIRIS), Institute National des Sciences Appliquées de Lyon, France, 7 avenue Jean Capelle, 69621 Villeurbanne cedex*

{ Addisalem.Negash, Vasile-Marian.Scuturici, Lionel.Brunie} @ insa-lyon.fr

ABSTRACT:

This paper presents SAMi, a context aware Information sharing Middleware, for a dynamic pervasive computing environment such as a Mobile Ad-hoc NETworks (MANET). SAMi can be used by co-located devices to share information according to the context of users and their environment. It applies latent semantic modeling techniques and adopts techniques from service discovery to search files efficiently. A preliminary prototype has been developed to simulate MANETs and to implement the middleware. A number of experiments have been performed to evaluate major features of SAMi. Based on these experiments, we found SAMi a promising middleware to facilitate information exchange among co-located devices.

KEYWORDS: Pervasive Computing, MANET, Information Sharing, latent semantic, Service Discovery

1. INTRODUCTION

A Mobile Ad-hoc NETwork (MANET) [9] is an important technology to support information exchange in pervasive computing environments. MANET is a collection of devices communicating with each other in the absence of any infrastructure. MANET can be formed in a battlefield by soldiers to exchange site information, enemy information and etc. It is also possible to form a MANET in a meeting room in order to facilitate exchange of information about historical places, schools, restaurants, or music. It can also be formed in a city bus, a shop, in a street, etc.

The focus of our research is to build information sharing system that allows users to exchange information wherever and whenever they got the opportunity through the use of a

MANET. Information sharing is vital regardless of location, time and computational environments. Despite of its importance, it has not been given enough focus. There are researchers who argue that information sharing can be treated as discovery and delivery of services since providing any kind of resources including information, hardware and software resources can be consider as a service. However, the environment is overloaded with information than any other kind of resources.

MANETs are heterogeneous in terms of nodes[1] mobility pattern, node density and users' interest. A mobility pattern of a node describes how fast a node moves and how long it stays at a certain place. It is high in the battlefield, i.e., users move fast and they don't stay long in a certain place. In the contrast, it is low in the meeting room since the meeting may require hours and users rarely move. A node density expresses the number of nodes per area. MANET formed at home will have only few devices in contrast to the one formed at a festival. MANETs are also different in terms of users' interests. The users at a MANET formed in the university will be interested to share academic information.

SAMi is a self adaptive middleware. It is used by co-located devices to share information transparently. SAMi is aimed to work in a MANET showing different characteristics in terms of nodes' mobility pattern, and density as well as users' interests.

The middleware divides environments according to the mobility pattern of nodes, that we call mobility zones. The middleware can, for example, consider the existence of three kinds of mobility zones which are created when nodes moves in fast, slow or moderate ways. SAMi shows different characteristics on different mobility zones of MANETs. The number and the characteristics of mobility zones are determined from devices' experience.

The middleware applies a vector space modeling techniques to have a common virtual information sharing space. The content of virtual information space is different for different mobility zones of MANET and varies according to users' interest. Most of the required computation needed to create this virtual space is computed offline.

The paper is organized as follows. Section 2 presents related works in the area of service discovery and information sharing. SAMi is presented in section 3. Section 4 discusses a preliminary prototype that has been used to simulate MANETs and to implement the middleware. We give conclusion and indicate the future works in section 5.

2. RELATED WORK

There are a few research works in the field of information sharing for MANET such as 7 DS [12] and ORION [10]. 7DS is designed to provide web browsing service to MANETs' users without connecting to Internet. The problem solved by 7 DS is quite different from ours. 7 DS allows users without Internet connection to access documents in the web with the help of others while our middleware allows co-located users to exchange information with each other. ORION is a file sharing system which works on top of overlay networks which is independent of the locations and the physical interconnections of peers.

Service discovery middleware (SDM) can be a base to design information sharing systems because providing information is a service. We can classify SDM into two categories: directory-based SDM [2,6,8, ,16,18] and directory-less SDM [1,3,4,7,11,13,14,17,20]. The

[1] We use node and device interchangeably

main difference between the middleware in the two categories is the existence of devices which provide a directory service in the former.

In directory based SDM, one or more devices provide directory services. Service-providers advertise their services to directories. To access a service, a client first contacts the directories to obtain the service descriptions in order to contact service providers directly. In a MANET, it is not always possible to get a device that can serve as a directory. Electing a directory and advertising the existence of a directory is costly.

In directory-less SDM, every device is required to provide some form of a directory (service registry). Each service provider advertises its services to devices in vicinity. Any device that is interested in a particular service(s) stores the advertisement in its local service registry. When a client wants a service it uses the cached advertisement to discover the service or it multicasts/broadcasts the service discovery message throughout the environment.

Service information can be updated whenever a certain event occurs such as unavailability of routs as in [1]. It can be also updates on regular basis for example by periodical advertisements as done by Konark [16], GSD [3 and 4], and middleware in [1, 8 and 15]. The period of advertisement can be variable according to the mobility pattern of nodes as it is done in Allia [14] and Konark [16].

The advertisement diameter is the number of hops that the advertisement traverses. In DeaPspace [11] and MoGATU [6], the advertisement diameter is one hop. In GSD [3 and 4] and Allia [14], diameter of advertisements is changed as the mobility pattern of node changes

We observe that the described information sharing and service discovery middleware are not self-adaptive and context aware, i.e., they don't adjust their behaviors according to the characteristics of a MANET and the context of users. Nevertheless, there are middleware which propose adjusting the diameter and period of advertisement as of the mobility pattern of devices. However, being self adaptive includes changes in the behavior of the middleware with regards to the content of advertisement and the number of devices used to deliver a single file (service) in addition to changes in the diameter and period of advertisement.

3. INFORMATION SHARING MIDDLEWARE

SAMi (Self-Adaptive Information sharing Middleware) allows devices to share information regardless of their mobility pattern and density. It considers the affinity of users or their common interest to select the information to be searched or provided. The middleware is composed of profile manager and information manger modules. Figure 1 shows the architecture of the middleware.

The middleware works proactively with the help of the profile manager by determining the information need of users. The information need of users is taken out from their agenda, and habit. The module makes the middleware flexible by determining the mobility pattern of users and their common interest from the user agenda and historical knowledge. The personal data-store contains data with related to a user including the user's agenda, profile, and habits.

Information manager module is responsible to manage the information sharing according to the users' mobility pattern and affinity (or common interest). This module is responsible to fetch and provide information to and from devices in the vicinity. It consists of two main modules: Advertisement Manager (AdvMng) and Information Discovery and Delivery manager (infDisDel). AdvMang (Advertisement Manager) is responsible to advertise

information according to mobility pattern of nodes and users' affinity. It is also responsible to manage the accepted advertisement. The InfDisDel (information discovery and delivery) module is responsible to manage file discover and delivery using one or more nodes. File discovery is done by using advertisements. The selection of the information providers to deliver a file is derived from the mobility pattern of nodes in the environment. This module is also responsible to handle queries of other nodes and to modify the history data- stores according to the accepted requests.

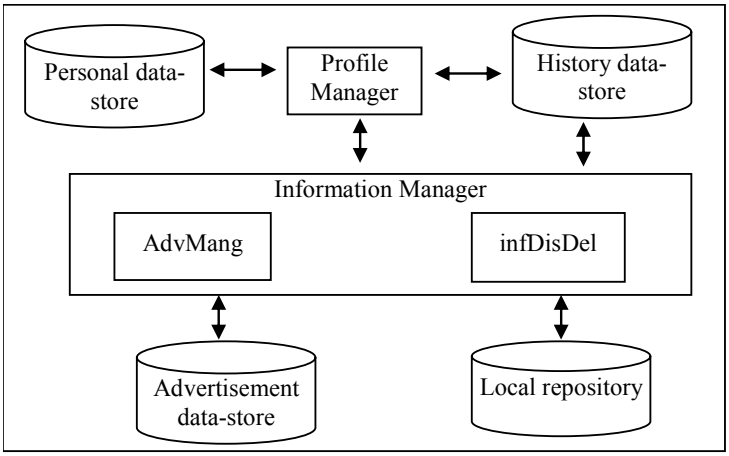

Figure 1. Architecture of SAMi

3.1 Profile Manager

The profile manager (figure 2) learns about the mobility pattern of nodes, and the information needs and affinity of users through the use of user's agenda, habit and profile as well as historical networks that the device was involved in.

Mobility pattern describes how long a user stays at a certain place and how fast he/she moves. Mobility pattern can be derived from the life time, agenda and habits of users and from networks that a node was involves in. Mobility pattern indicates to which mobility zones a node is in. Mobility zones are mobile ad-hoc environments having a specific mobility pattern. We can have fast, slow or moderate mobility zones. A life time of nodes, i.e., how long nodes stay at a certain place, is an important input to estimate their mobility pattern. A life time of a node can be predicted from historical knowledge of networks that a node was involved in and from a user's agenda and habit. For the user whose information is displayed in figure 3, the module can estimate the user's life time as 30 minutes when he is doing shopping on a weekend. If the same user involves in a MANET at 10 A.M, his life time is calculated from his agenda as 2 hr.

A device can estimate the characteristics of a MANET from its historical knowledge by analyzing data found in historical data-store. From historical data-store, a device can learn a MANET in the meeting room has slow mobility in contrast to the one in city bus.

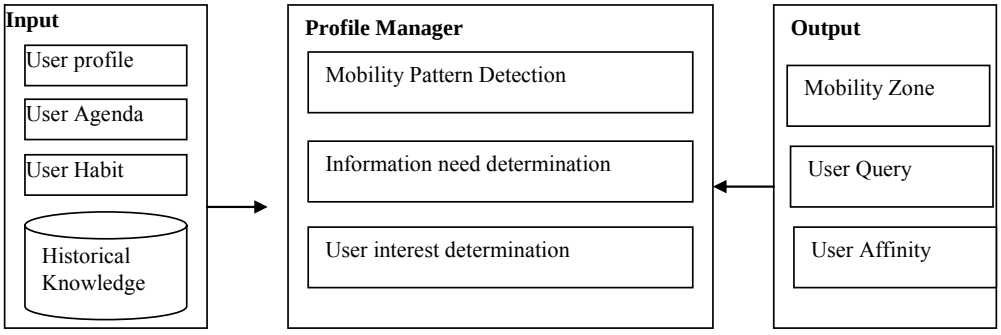

Figure 2. The profile manager module

If a device is unable to determine how the neighbors move from the historical knowledge and user's agenda and habit, the mobility pattern can be determined from the time to live (TTL^2) of all devices in the environment. Therefore, devices can know the mobility pattern of each other by exchanging their time to lives. This will, however, create unnecessary network traffic. In SAMi, if it is not possible to know the mobility pattern of nodes, a device will assume that it is in a dynamic environment and learn about the exact behavior of the environment eventually.

Profile manager examines a user's agenda and habit to predict what kind of documents a user wants to have. The information need of a user having agenda described in figure 3 includes documents about "Steps of strategic plan preparation", "Efficient way of chairing a meeting" and "How to deal with business man". For user who has a habit of listening music during a long journey (e.g., user in figure 3) with preference of Whitney's songs, if he has such a plan, the device starts searching the mentioned music.

Profile Manager detects the common interests of users using their profile and agenda as well as by analyzing the data found in historical data-store. The user profile can indicate whether a user comes from the same country or are in the same profession or have the same hobbies. User affinity can also be derived from historical data. From history, one can learn the user affinity of MANETs formed in stadium is to share information related to sport.

[2] we use life time and time to leave interchangeably

User Agenda			
Start	**End**	**Activity**	**Required Documents**
10A.M.	12 P.M.	Strategic plan preparation	Steps of strategic plan preparation
12P.M.	1 P.M.	Lunch	
1P.M.	3 P.M	Meeting with business MAN	Efficient way of chairing a meeting How to deal with business MAN

User Habit			
Habit Name	**When**	**Time needed**	**Activity**
Shopping	Week end	30 minutes	-
Journey in train	Friday	2 hours	Listen music
Talking with friend	Night	10 minutes	Exchange jokes
	day	2 minutes	-

Historical MANETs				
Place	**Time**	**File exchanged**	**Mobility zone**	**User affinity**
Bus	8A.M.–8:30A.M.	Foot ball, museum	fast	
Campus	12:30 P.M–1 PM	Research paper	moderate	students
Seminars	2 P.M–4 P.M	Academic profile, C.V., research papers	slow	The same profession

Figure 3. Examples of historical data and user agenda and habit

3.2 INFORMATION REPRESENTATION

A device prepares a metadata for each sharable file as in Figure 4. A metadata of a file consists of the identifier, the name, the size and the time to leave of a file as well as keywords to describe its content.

File-ID	keywords	Format		TTL
		type	size	

Figure 4. file description

To facilitate file searching and categorization, files are mapped in a space using vector space modeling techniques (VSM)[5,15]. Vector space modeling is a standard technique in information retrieval to represent documents through words they contain. The technique will be used to facilitate searching a document and to identify similarity between documents. In the middleware, vector spaces are used to cluster and retrieve files. Clustering is done statically by considering possible mobility patterns of nodes. We introduce a concept of virtual vector space to represent important files in the network to facilitate file searching.

Doc/Term	T1	…	Tm
D1	w_{11}		w_{1m}
..			
Dn	w_{n1}		w_{nm}

Figure 5. Term-document frequency matrix

In VSM, documents are represented by a vector di={w_{i1},w_{i2}, . . ., w_{in}} where w_{ij} represents the weight of term j in document i. To produce this vector, the documents are parsed into series of words in such a way that the parsing process removes stop words such as prepositions, conjunctions, common verbs, pronouns, articles and common adjectives. The documents are then represented in term document frequency matrix as shown in figure 5.

Category -Name	keywords	child		TTL
		Number of children Files	Number of children Categories	

Figure 6. File category description

The document vector can be considered as a point in the nxm space. The dimension of the space can be reduced by applying singular vector decomposition. The document vector is termed as a vector space. After documents are represented in a vector space, they will be classified hierarchically. Files are first classified into different categories by using k-means clustering algorithm [19, 21]. Further categorization can be done if there are several categories. Categorization will continue and they form a tree. Figure 7 shows an example tree created by file categorization. Each category has a name which can speak about files in it and is described as a sequence of keywords. A category description is derived from the descriptions of files and sub categories found under that category. The metadata of file categories are displayed in Figure 6.

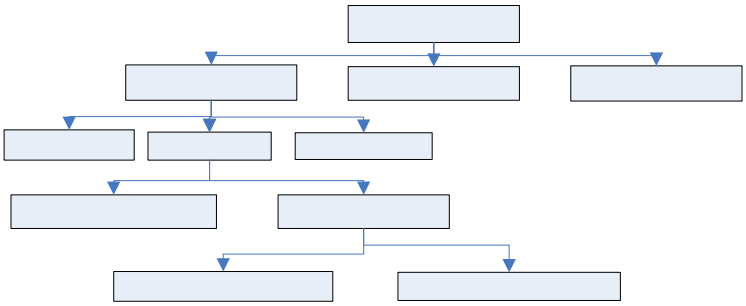

Figure 7. Examples of information grouping

Classification is done by applying k-means algorithm repeatedly. K-means algorithm works as follow. K points which represent files in the vector space are selected randomly. These points become initial clusters' centers. The files are mapped to clusters. A file is mapped to a cluster if the cosine of the angle between point representing the file and cluster's center is less than to any of the angles between the file and other clusters' centers. After mapping is finished, the center of the clusters will be recalculating from the point representations of files under them. File mapping and cluster center calculation is done repeatedly until the centers are no more changed. The resulting clusters will be represented by their centers. Using the same algorithm, clusters are further classified. Further classification is repeatedly done until a tree of required depth is obtained. The root cluster is nothing but an artificial cluster which represents all the files on a device.

Representation and classification of file are performed offline, i.e., before a device enters into a MANET, to overcome the problem imposed by the running time requirement of VSM techniques.

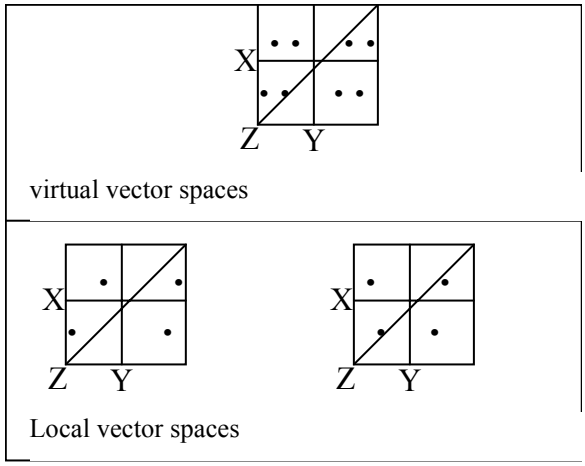

Figure 8. Constructing virtual vector space by mixing local vector spaces

Virtual vector space is constructed to represent the accepted advertisements. As shown in figure 8, a simple way of constructing a virtual vector space is by mixing the local vector space which is constructed using latent semantic indexing [15]. This, however, requires exchange of terms and construction/modification of vector space as one node enters into the network. Construction or modification is too expensive to perform in a MANET where devices enter and leave frequently. In SAMi, a virtual vector space is constructed from the meta-data of files and clusters. Virtual vector spaces are constructed by using the techniques that are used to construct local vector space (vector space for local documents).

3.3 RESOURCE ADVERTISEMENT

Each device advertises files or/and clusters. The advertisement message can contain only clusters found at the shallowest or the deepest level of the file-tree. The content of advertisements is determined by mobility patterns of nodes and user affinity. The diameter and frequency of the advertisements are determined according to the mobility pattern of the nodes. The advertisement diameter is the number of hops that the advertisement traverses. A hop describes how the devices are interconnected. One hop neighbor of a node is the one to which the node has a direct connection. Two hop neighbor of a node is the one which has a direct connection to one of the node's one hop neighbors.

Mobility pattern is inversely proportional to the number of hops and content of advertisement and directly proportional to the frequency of advertisement. If the nodes are highly mobile, the number of hops and the content of advertisement, i.e., the number of files and clusters to be advertised, should be small and advertisement should be done frequently.

From historical knowledge, a node can learn that the advertisement diameter should be one hop for a MANET in a city bus and can cover the whole network for the one formed in the meeting room. The historical data can indicate the need of doing advertisement frequently for the former but occasionally for the latter.

Number of files/clusters to be advertised is determined according to the mobility zone a node is in. From experience, advertisement manager can learn about possible mobile zones of MANET, there characteristics and the number of files/clusters advertised for each of the mobile zones as shown in table 1. The selection of files for advertisement is determined

according to user affinity. If all the users like football, files about football and sport should be given priority.

User Affinity is described as a set of users' interests. The user interest I is described by a set of keywords. The set, {football, tennis, basketball} can describe the interests of users who like sports. There may be users who don't have any special interest. The middleware represents the interest of these users as empty set. There may also be users who have two or more interests. For example, a user may be interested to share files about music and football. Therefore, each interest I is attached with affinity value that indicates how many of the users are interested on I.

Table 1. Mobility Zones

Zone	Life Time	Advertisement
1	Up to LT_1	$Nadv_1$ Files/Clusters
2	LT_1 to LT_2	$Nadv_2$ Files/Clusters
3	LT_2 to LT_3	$Nadv_3$ Files/Clusters
. . .	. . .	. . .
N	LT_{N-1} to LT_N	$Nadv_n$ Files/Clusters

The advertisement manager maps all user interest except an empty interest in the vector space. For each interest I, it puts files f and Cluster c in a sets F(I) and C(I) respectively iff c and f are go with interest I. The closeness of files and clusters are determined using definition 1. For the vector space displayed in figure 9, F(I) = {f5,f9, f10,f11,f12,f13} and C(I) = {c3,c4}

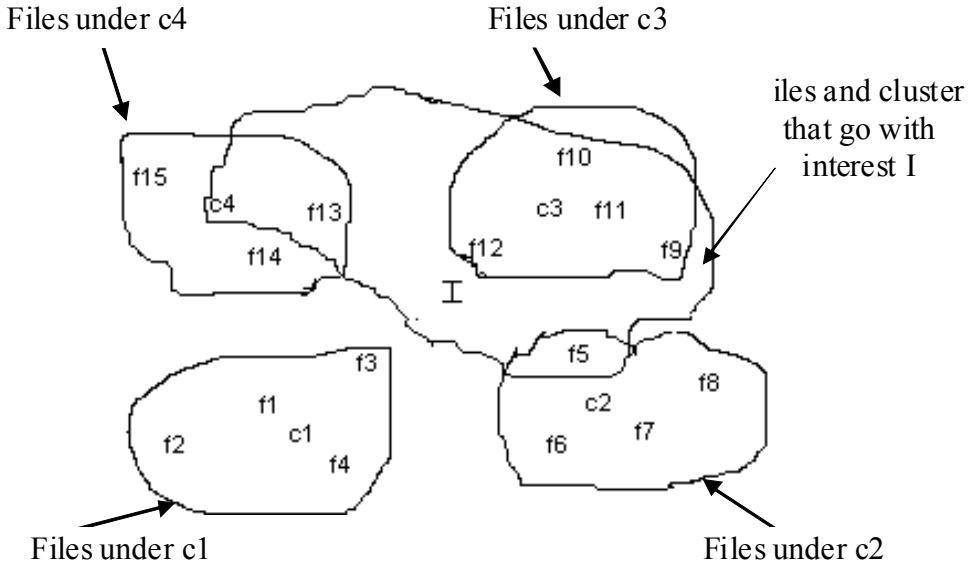

Figure 9. Interest mapping example

The following symbols are used in definitions that we are going to state in this section.

- Fpi is a point representing file fi,
- Cpi is a point representation cluster ci,
- Ipi is a point representing interest Ii,
- Θij is an angle between Fpi and Ipi,
- γij is an angle between Cpi and Ipi, and
- α is a similarity threshold of a file and an interest

Definition 1: a file fi is go with interest Ij iff cosine (Θij) > α and cosine (Θij) > cosine (Θik) for k # i.

Definition 2: a file fi is more relevant to interest Ij than a file fk iff cosine(Θij) > cosine(Θkj).

The above definitions work for clusters by replacing Θ by γ and f by c.

If there is an empty interest I, the advertisement manager puts all files f and cluster c which are not found in non of F(Ii) and C(Ii) in F(I) and C(I) respectively.

The advertisement manager selects files and clusters from F(I) and C(I) for each interest I using Algorithm 1. The algorithm determines the files/clusters to be advertised for each interest Ii.

The following set is used by the algorithm 1.

- M(f) and M(c) are the metadata of file f and cluster c respectively (see section 3.2 for metadata representations of files and clusters).
- FC(I) is a set of files which are in F(I) and classified under some cluster c in C(I). For the example in figure 9, FC(I) = { f9, f10,f11,f12,f13}.
- FNC(I) is a set of files which are in F(I) but not in FC(I), for the example in figure 9, FNC(I) = {f5}
- ADV is a set of metadata of files and clusters to be advertised,
- N(I) is the number of files and clusters advertised with related to interest I, and
- NR(I) is the remaining number of files and clusters to be advertised with related to interest I.

The advertisement quota for interest I, i.e., the number of files/clusters to be advertised according to each interest I, is calculated from the equation N(I) = P(I)Nadv where P(I) and N(I) are the percentage of users interested on an interest I and the number of files or clusters can be advertised for an interest I respectively.

The algorithm selects the number of files to be advertised for interest I as follows. We assume that N (I) is less than or equal to F(I). If N(I) is enough for advertising all file in F(I), it only advertises files in F(I) and ignores the clusters in C(I). If the opposite happens, it give priority to clusters in C(I) and then files which are in F(I) but not classified under any cluster c in C(I). In cases where N(I) is less than |C(I)|, advertisement manger selects the more relevant elements of C(I). For the interest I shown in figure 9, if N(I) is 1, only c3 will be selected. The case where N(I) is greater than C(I), is treated by taking all the clusters and files which are in F(I) but not classified under any clusters in C(I). If N(I) is 3, c3, c4 and f5 will be selected. If some quota is rested after processing the above step, the advertisement manager adds more relevant among files which are in F (I) and classified some cluster c in C (I). If N(I) is 4, c3, c4, f5 and f2 will be selected. More relevant files and clusters are identified using definition 2.

```
If N(I) == |F(I)|
    Put M(f) in ADV for all f € F(I)
Else if N(I) <= |C(I)|
    Select the N(I) most relevant clusters from C(I) and put their metadata in ADV
Else
    Put M(c) in ADV for all c € C (I)
    NR(I) = N(I) - |C(I)|
    If (NR(I) <|FNC (I)|)
        Select the NR (I) most relevant files from FNC (I) and put their metadata in ADV
    Else
        Put all M (f) in ADV for all f € FNC (I)
        NR(I) = NR(I) - |FNC (I)|
        Select NR(I) more relevant files from FC(I) and put their metadata in ADV
    End If
End if
```

Algorithm 1. Select files and clusters for advertisement

The content of ADV is advertised to selected nodes in the neighborhood. The node selection is done according to the mobility zone where the node is involved in. The node can decide to advertise to its direct neighbors if it is in highly dynamic zones and to every node if it is in zones where nodes rarely move.

Advertisement should be performed repeatedly because of the dynamicity of MANETs. For this, it is important to decide the period of advertisement. The period of advertisement can be determined from the life time of users. However, it is also be possible to make the content of advertisement in the next period a little bit different by giving priority to files which have not been advertised before.

3.4 INFORMATION DISCOVERY AND DELIVERY

Information discovery is performed according to the information need of a user. First, the user information need is described as a list of keywords and then, the needed information is searched from the virtual vector space that represents all the files and clusters advertised.

A user query, Q, is expressed as a list of keywords. Q is, then, mapped in the virtual vector space. Files f and cluster c are placed in F (Q) and C (Q) respectively if they are relevant to query Q. Definition 3 is used to decide whether files or clusters are relevant to a query.

The following symbols are used in the definitions that we are going to state in this section.

- Fpi is a point representing file fi,
- Cpi is a point representation cluster ci
- Qpi is a point representing query Qi
- Θij is an angle between Fpi and Qpj,
- γij is an angle between Cpi and Qpj, and
- α is a similarity threshold of a file and a query

Definition 3: a file fi is relevant to a query Qj iff cosine (Θij) > α.

Definition 4: a file fi is more relevant to interest Qj than a file fk iff cosine (Θij) > cosine (Θkj).

The above definitions work for clusters by replacing Θ by γ and f by c.

Algorithm 2 is used to prepare discovery and delivery for query Q. Algorithm 2 decides files to be delivered and discovered from F(Q) and C(Q) respectively. The number of files to be discovered, n, is determined from nodes' experience.

The following functions and sets will be used in algorithm 2

- BSF(Q) and BSC(Q) contains the most relevant files and clusters in F(Q) and C(Q) respectively. Definition 4 is used to decide if a file or a cluster is more relevant than any other files or clusters.
- LifeTime(D) is a function that outputs the lifetime of a device
- Owners(f) and Owners(c) are sets that contains all owners of f and c respectively
- DownloadTime(f) is the time needed to download a file f.
- DiscoveryDelivery(c) is the time needed to discover and deliver one file categorized under a cluster c.
- Discovery (Q) is a set of devices that a discovery message will be set to.
- Delivery (Q) is a set of delivery messages. A delivery message includes a file and device to deliver the file.

Algorithm 2 works as follows. It first removes files and clusters from F (Q) and C (Q) if it is not possible to deliver or discover files before the owners of files or clusters move. If |F(Q)| is greater than n, it considers the n most relevant file in F(Q) for file delivery. In the case where |F(Q)| < n, it considers all the files in [F(Q)| for file delivery and takes all or some of the clusters in C(Q) to prepare discovery messages. The algorithm considers the owner of the most |C(Q)|-n-|F(Q)| clusters in devices for file delivery if |C(Q)| >n-F(Q). It considers the owners of C(Q) for file discovery for the opposite case.

If enough advertisements about files are found, the delivery phase can be started automatically. If that is not the case, discovery message will be sent to the owners of the clusters in BSC (Q). When a device receives a query Q, it searches a file that matches with Q. The device then sends the file description to the requester peer if the search is successful. Information discovery phase is finished when the deadline of file discovery is expired or/and enough information providers are discovered.

After the information discovery phase is completed, the information delivery phase will be started. The purpose of this phase is to select one or more information sources to deliver files that match a query Q. Information source selection is based on how far information-sources are found from the requester in terms of distance and number of hops. The information delivery is performed as follows:

1. For each discovered file, SAMi searches information-sources which can deliver the whole file. The node knows if a source can deliver the whole file or not from the TTL of information source and file size. If there are several such types of information source, SAMi selects an information source according to how far it is and how long it will stay around.
2. If the search in the above step is not successful, SAMi searches a combination of nodes (n_1,n_2, …, n_k) for some integer k such that information source n_i delivers a portion of a file called f_i and the merge of f_i gives the required file.

Fault can occur either in information discovery or delivery phases. A node estimates the maximum time required to discover a certain file. Information discovery fault occurs if the node doesn't discover a source for that file within the estimated time. In that case, another discovery message for similar files in C (Q) – BSF (Q) will be prepared. If C (Q) –BSF(Q) is empty, it waits until it gets an advertisement related to Q.

When a node asks information source to deliver a file or a portion of it, it calculates the maximum time to get the information. If the source doesn't get the information within that time, it concludes that information delivery has failed. In this case, it searches if there are other information sources to deliver the required file or portion.

```
Remove any f in F(Q) where ∑lifetime(di) for di € owners(f) < DownloadTime(f)
Remove any c in C(Q) where ∑lifetime(di) for di € owners(c) < DiscoveryDelivery(c)
If (n <=|F(Q)|)
    Put the most n relevant files of F(Q) in BSF(Q)
    For each f € BSF(Q)
        For  d € owners(f)
                Put ((d,f) in Delivery(Q)
        End For
    End For
Else
    For each f € F(Q)
        For each d € owners(f)
                Put ((d,f) in Delivery(Q)
        End For
    End For
    Put the most n - |F (Q)| relevant clusters of C (Q) in BSC (Q)
    For each c € BSC(Q)
        For each d € owners(c)
                Put ((d,Q) in Discovery(Q)
        End for
    End For
End If
```

Algorithm 2. prepare delivery and discovery message for a query

4. SIMULATION

A test-bed depicted in Figure 10 has been developed to simulate a MANET and to implement the proposed middleware. The test-bed accepts the following inputs.

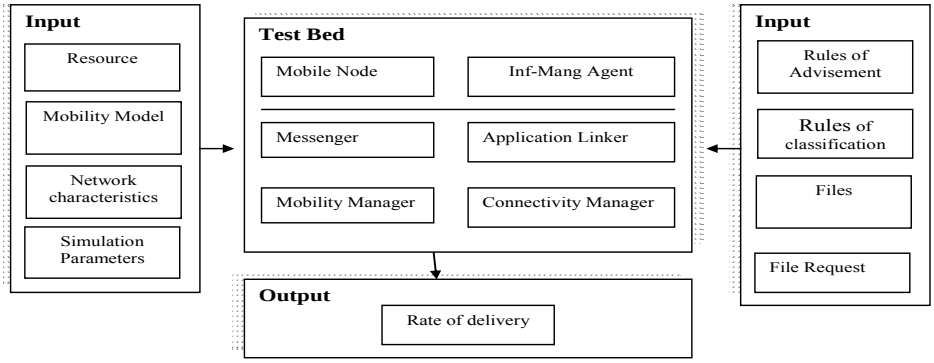

Figure 10. A Test bed to simulate a MANET

*Resources***:** consists of the properties of important of devices' resources such as CPU and memory. The current implementation only accepts memory capacity but the test bed is open to include other resources as CPU.

Files: represents abstracts of research papers.

File requests: are set of queries which are a list of keywords. Each query is attached with a time to search and a deadline to deliver files for the query.

Simulation Parameters: consists of the area coverage of the MANET, a simulation unit and duration of a simulation.

Mobility Model: The test-bed implements a random way point distribution model to determine the distribution of mobile nodes and how frequently they move. It only accepts the maximum and the minimum values for the speed and the pause time of the nodes. A pause time of a node is a time that the node stays at a certain place.

Network characteristics: The bandwidth and the line of sight of the network technology that are used to connect two nodes.

Rules of advertisement: dictates the number of mobile zones and the maximum number of file or cluster advertisement for different mobile zones.

Rules of classification: indicates the depth of the file tree and the number of clusters at each level of the tree.

The output of the test-bed presents the rate of file delivery which is the ratio of number of delivered files to the number of requested files.

The test bed has two components for simulating the network and implementing the middleware. *Inf-Manag agent* and *Mobile node* modules are used to implement the middleware.

Mobile node simulates mobile devices. It has hardware, software part and data parts. The *h*ardware part includes Memory and CPU. Inf_Manag agent is considered as the software component of the device. Files and the information need of the user are considered as the data part.

Inf-Manag agent is the software component of the mobile device. It is used to simulate information manager of SAMi. It makes classified files hierarchically by accepting rule of classification. It makes advertisement according to the rule of advertisement. It is also responsible to discovers and delivers files for a file request.

For simulating the network, it provides the following modules.

Messenger: It is responsible for messages transaction. It accepts the message M from Inf-Mang-Agent through application linker. M can represent an advertisement, a file request, or a

file. It divides M into packets MPi. For each destination node of M, it calculates when to send Mpi. Mpi is sent when the time comes.

Connectivity Manager: It is in charge of checking the connectivity between two devices according to the characteristics of the communication technology used. In each simulation unit, connectivity manager checks if the new connection between devices is needed. If that is the case, it makes them interconnected. It makes node disconnected if the connection between them is no more valid.

Mobility Manager: It is responsible to the position and the speed of each device according to the mobility model used.

Application Linker: It is responsible to invoke application when events specified by an application are occurred.

Table 2. the inputs of the test-bed

Parameters	Values
Simulation area	33 meters by 33 meters
Simulation time (STime)	10800 second
Simulation unit	Nanosecond
Network bandwidth	720 Kbps
Line of sight	10 meters
Number of files	10*Number of Nodes
Number of replication for each file	Random (0,8)
Request Time (RT)	Random (0, STime)
Deadline for file delivery	Random(RT+1200s, STime)
Number of file request per node	Random (0,number of files)
Meta-data size	Random(1KB, 4KB)
Storage capacity of a node	Random(256MB, 160GB)
Speed	Random(1m/s,45m/s)

The middleware was tested by fixing the simulation area, simulation time and file density. We searched 500 papers in the domain of information retrieval, multimedia, pervasive computing, GIS and e-learning. The abstracts of those papers are taken as files in the experiment. For each file at least there is a query that matches with the file. The files and queries are distributed onto the nodes randomly. Twenty files and queries were stored on each node. Two or more nodes might own identical files, i.e., the files are replicated. The node, however, doesn't keep the file that it has got from others as sharable file. Table 2 shows the inputs of the test-bed during the experiment.

Three types of mobile ad-hoc environments which are highly dynamic, dynamic and moderate were considered and displayed in table 3. The classification was done according to the life time of devices in the environment. We made two experiments for each environment by changing the number of nodes from 10 to 20.

Table 3.Types of Environments

Environment	Life Time Ranges(in minutes)	Number of advertisement
Highly dynamic	1 to 5	5
Dynamic	5 to 10	10
moderate	10 to 15	16

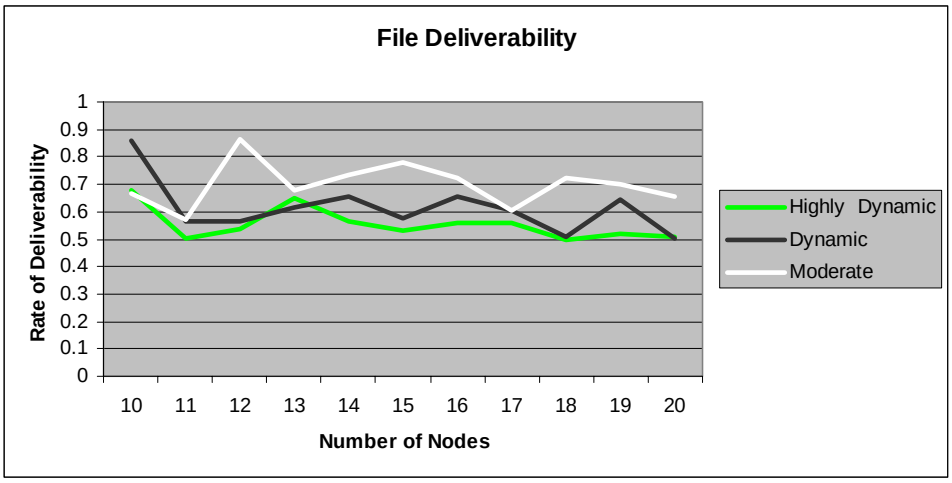

Figure 11. Deliverability of Files for experiment one

The figure 11 and 12 show the result of experiment one and two respectively. The difference between the two experiments lies on how file classification is done. For the first experiment, files are classified into 6 and then into 3 clusters. On the second experiment, files are classified into 10 and then into 5 clusters.

All requested files were not delivered because of one of the three reasons. One, the requested file might be found in the node which was far from the requester peer. The second reason is that the information source might disappear before the file was completely delivered and the information source and the requester peer didn't meet again. Third, the deadline of file delivery was reached before it was completely delivered.

As it is observed in figure 11 and 12, the rate of delivery, i.e., the number of files delivered out of the number of files that has been requested, shows similar pattern regardless of the environmental changes. The balance on the rate of file deliveries on highly dynamic, dynamic and moderate zones is created by the fact that the middleware uses three or more data sources to deliver a file in the case where a single node can't deliver the whole file. Moreover, different portion of a file can be delivered in different time.

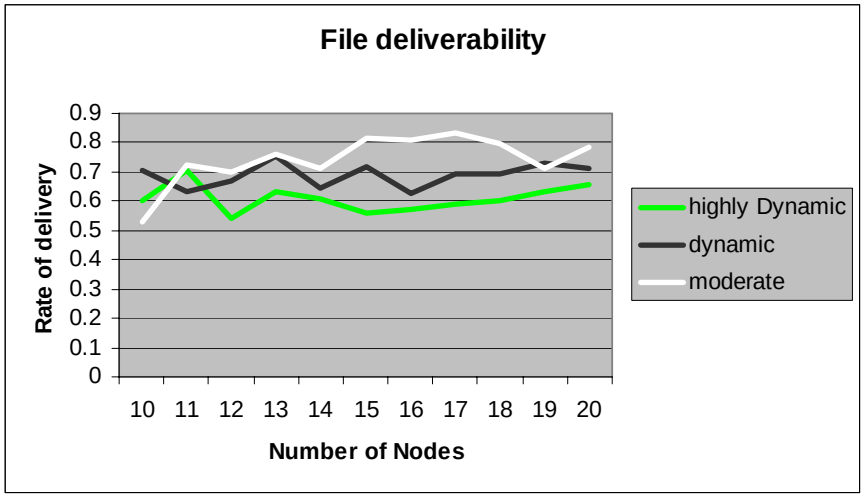

Figure 12.Deliverability of Files for experiment two

5. CONCLUSION AND FUTURE WORK

The paper proposes a self-adaptive middleware (SAMi) that uses a novel idea to enable co-located devices to share information with regardless of their mobility pattern and density. SAMi is an advertisement based middleware where the content, the diameter and the frequency of advertisement are determined according to the mobility pattern of nodes in the vicinity as well as the context of the users and their environment. In the middleware, the information exchange is done proactively. The information need of the user is estimated from his agenda and habit. Files are represented in vector space and are categories hierarchically in order to make the discovery process facilitated and to minimize the impact of advertisements on the bandwidth. We have developed a preliminary prototype to simulate a MANET and to implement the middleware SAMi. We have evaluated the prototype on different mobility setting and the result shows that SAMi is a promising middleware for sharing information in MANETs.

In future, we will extend the prototype to consider different network technologies and to consider users interests. We will also perform further study on construction and modification of virtual vector space.

REFERENCES

[1] Alex Varshavsky, Bradley Reid and Eyal de Lara. A cross layer approach to service discovery and selection in MANET. January 2004.

[2] Conte, M. Fredj, I. Hassine, J-P. Giraudin, D. Rieu, AGAP: an environment to design and apply patterns, IEEE International Conference on Systems, Man and Cybernetics (IEEE SMC), Hammamet, Tunisia, October 6-9, 2002.

[3] D. Chakraborty, A. Joshi, T. Finin and Y. Yesha. GSD: A .novel group based service discovery protocol for MANET. In 4rth IEEE Conference on Mobile and Wireless Communications Networks (MWCN). IEEE, September 2002.

[4] D. Chakraborty, A. Joshi, Y. Yesha and T. Finin. Toward distributed service discovery in pervasive computing environments. IEEE Transactions on Mobile Computing, February 2006.

[5] Erica Chisholm and Tamara G. Kolda. New Term Weighting Formulas for the Vector Space, Oak Ridge National Labs, 1999.

[6] Feng Zhu, Matt Mutka and Lionel Ni. Splendor: A secure, private and location-aware service discovery protocol supporting mobile services. Proceedings of the First International Conference on Pervasive Computing and Communication PerCom`03, Pages 235-242, 2003. ACM Press.

[7] Filip Perich, Anupam Joshi, Timothy Finin, and Yelena Yesha. On Data Management in Pervasive Computing Environments, IEEE Transactions on Knowledge and Data Engineering May 31, 2004.

[8] Jerry Tyan and Qusay H. Mahmoud. A network layer based architecture for service discovery in mobile ad hoc networks. CCECE 2004 , May 2004 IEEE.

[9] Jun-Zhao Sun. Mobile Ad Hoc Networking: An Essential Technology for Pervasive Computing, Info-tech and Info-net, 2001. Proceedings. ICII 2001 - Beijing. 2001 International Conférences on, 2001

[10] Klemm, C. Lindemann, and O. P. Waldhorst. A special-purpose peer-to-peer file sharing system for mobile ad hoc networks. In VTC, 2003.

[11] M. Nidd. “Service Discovery in DEAPSpace” IEEE Personal Comm., August 2001, pp. 39–45.

[12] M. Papadopouli and H. Schulzrinne. A Performance Analysis of 7DS, A Peer-to-Peer Data Dissemination and Prefetching Tool for Mobile Users. In Advances in wired and wireless communications, IEEE Sarnoff Symposium Digest, March 2001.

[13] L. Cheng and I. Marsic. Service discovery and invocation for mobile ad hoc networked appliances, December 2000.

[14] Olga Vladi Ratsimor, Dipanjan Chakraborty, Anupam Joshi, Timothy Finin. Allia: Alliance-based service discovery for ad hoc environments. In ACM Workshop on Mobile Commerce WMC`02, September 2002.

[15] S. Deerwester, Susan Dumais, G. W. Furnas, T. K. Landauer, R. Harshman. Indexing by Latent Semantic Analysis, Journal of the Society for Information Science 41 (6): 391-407, 1990

[16] Sailhan F. and Issarny V. Scalable service discovery for MANET. In proc of IEEE International Conference on Pervasive Computing and Communications (PERCOM), March 2005.

[17] S. Helal, N. Desai, V. Verma and C. Lee. Konark- a service discovery and delivery protocol for ad hoc networks. Proceeding of the Third IEEE Conference on Wireless Communication Networks WCNC, March 2003.

[18] Ulas C. Kozart and Leandos Tassiulas. Network layer support for service discovery in mobile ad-hoc networks. Proceeding of IEEE/INFOCOM-2003, April 2003.

[19] T. Kanungo, D. M. Mount, N. Netanyahu, C. Piatko, R. Silverman, and A. Y. Wu, An efficient k-means clustering algorithm: Analysis and implementation, IEEE Trans. Pattern Analysis and Machine Intelligence, 2002.

[20] Vincent Lenders, Martin May and Berhard Plattner. Service discovery in mobile ad hoc networks: A field theoretic approach. Pervasive and Mobile Computing 2005.

[21] Y. Zhao and G. Karypis. Empirical and theoretical comparisons of selected criterion functions for document clustering. In Journal of Machine Learning, Volume 55, Issue 3, page 311-331, 2004.