Proxy-based Middleware for Service Continuity in Mobile Ad Hoc Networks

Paolo Bellavista, Antonio Corradi, Eugenio Magistretti

***Abstract*—Advances in device miniaturization and wireless technologies are stimulating novel kinds of networks (Mobile Ad hoc NETworks - MANET) capable of autonomous peer-to-peer organization without the need of a statically deployed support infrastructure. MANET are specifically characterized by high mobility of network nodes and frequent changes of direct visibility. High dynamicity affects the design and implementation of distributed applications by significantly increasing their complexity, to consider not only routing and node configuration issues, but also the possible mobility of software components and the loss of direct connectivity during service provisioning. The paper proposes a highly dynamic and flexible middleware to support service continuity over MANET, i.e., to enable the continuous provisioning of a service, based on the client/server model of interaction, even if clients/servers move at runtime. The middleware is based on application-transparent proxies that act as decoupling intermediaries between the moving clients and servers. The proxy role is assigned dynamically in a completely decentralized way. Proxies exploit code mobility to install only when and where needed. The paper aims to show how the middleware facilitates the development of applications with service continuity, by describing the design and implementation of a file transfer case study. First experimental results demonstrate the feasibility and effectiveness of our approach at the application level and point out the suitability of mobile code programming paradigms in highly dynamic MANET environments.**

***Index Terms*—Middleware, Mobile Ad Hoc Networks, Mobile Code, Proxy, Service Continuity.**

I. Introduction

WIRELESS systems have recently become more and more popular, mainly because of their offered capability of supporting continuous mobility while accessing services. The spreading of wireless solutions is significantly changing also the way to conceive and design distributed services. On the one hand, the new features introduced by the wireless network infrastructure suggest application developers to create location-aware services [1]. On the other hand, the necessity of rapid, flexible and temporary connections between heterogeneous wireless devices is motivating the research for Mobile Ad hoc Networks (MANET). Any node in a MANET can move at any time; therefore, topological variations force the continuous reorganization of the network, which must occur in an autonomous, spontaneous and transparent way [2]. MANET are capable of operating without infrastructure support, because each node is autonomous and can collaborate with the others to enable information delivery.

Designing and implementing distributed applications for MANET is significantly more complex than for traditional fixed network environments. In particular, MANET high dynamicity forces service developers to face both issues introduced by the novel network properties: infrastructure lack and terminal mobility. Because of these two characteristics, even the client/server model should be reexamined in the MANET deployment scenario. In fact, the infrastructure lack requires reconsidering issues such as routing, loss of connectivity, and connection reliability, typically delegated and (at least partially) solved by the network infrastructure. In addition, frequent terminal mobility requires clients to perform continuous discovery operations to update the list of devices/services in network visibility, and to operate the needed client-to-server rebinding accordingly. Let us note that most available solutions for device/service discovery are based on a quite static infrastructure, for instance of lookup servers hosted in fixed network nodes, and are not designed to deal with the usual movement of clients/servers during both service discovery and provisioning.

Given above the motivations, we have designed and implemented a highly dynamic and flexible middleware to support *service continuity* over MANET. The middleware should enable the same continuous service provisioning provided by the traditional client/server model of interaction, even for mobile clients/server. The middleware operates at the application level, to achieve high flexibility and full portability over different MANET implementation solutions, from heterogeneous wireless connectivity technologies (IEEE 802.11, Bluetooth, …) to different multi-hop routing protocols. The paper claims that the application level is the most suitable one to provide flexible solutions to crucial mobility issues, such as security and interoperability; application-level middlewares can relevantly benefit from the availability of standard mechanisms, solutions, and tools at

Manuscript received May 14, 2003.

P. Bellavista is with the Dipartimento di Elettronica, Informatica e Sistemistica (DEIS), Bologna, IT (phone: 039-0512093866; fax: 039-0512093073; e-mail: pbellavista@deis.unibo.it).

A. Corradi is with the Dipartimento di Elettronica, Informatica e Sistemistica (DEIS), Bologna, IT (e-mail: acorradi@deis.unibo.it).

E. Magistretti collaborates with the Dipartimento di Elettronica, Informatica e Sistemistica (DEIS), Bologna, IT (e-mail: emagistretti@deis.unibo.it).

this abstraction layer [3].

The proposed middleware is based on the idea of application-transparent proxies that act as decoupling agents between client and server endpoints in order to support their binding/rebinding independently of mutual movements during service delivery. However, due to the intrinsic lack of infrastructure in MANET and to their high dynamicity, we have decided to assign dynamically the role of proxy agents among the peers in a completely decentralized way via an election protocol. Only when a server exits a locality during a service session, the middleware transparently reorganizes the roles in the involved locality and triggers the proxy election, if needed. The designated proxy is in charge of transparently looking for a suitable server component, either local or remote, and of forwarding to it the client requests, which are automatically directed to the local proxy by the middleware. The proxies exploit code mobility to install their behavior and the needed routing protocols only when and where needed.

Finally, the paper presents the design and implementation of a case study, the File Transfer in Mobile Ad hoc Networks (FT-MAN) service prototype, that exemplifies how the proposed middleware significantly eases the development of applications with service continuity over MANET. The reported experimental results show the feasibility and effectiveness of our approach at the application level and point out the suitability and flexibility of mobile code programming paradigms in the addressed deployment scenarios characterized by high dynamicity.

The remainder of the paper is structured as follows. Section II provides a rapid overview of MANET, of the main issues in developing client/server services over them, and of the state-of-the-art of the literature in the field. Section III presents the architecture and the design guidelines of our middleware to support service continuity in MANET, while Section IV describes how to build the FT-MAN application on top of the proposed middleware. Section V is devoted to present the first performance results obtained by deploying both the middleware and FT-MAN over IEEE 802.11b devices that exploit the ad hoc connectivity mode. Conclusions and on-going research work end the paper.

II. MOBILE AD HOC NETWORKS: OVERVIEW AND OPEN ISSUES

MANET identify a specific type of wireless network without requiring any kind of statically deployed support infrastructure, but permitting any node autonomy and cooperation to service delivery by forwarding messages along multi-hop paths. MANET are based upon autonomy and fast deployment, at the cost of continuous re-organizations due to frequent and unpredictable node movements. MANET applications can be profitably deployed in several different environments, from hostile grounds/disaster-recovering scenarios where a network infrastructure typically does not exist or has been destroyed (military and search-and-rescue operations), to contexts where the rapidity of the network deployment process is paramount (during a conference in a convention hall). It is possible to identify three different classes of MANET, with different degrees of complexity, by considering the physical dimensions and the number of participating nodes: sensor networks are low power, low range, and suit simple monitoring operations, e.g., on buildings and transportation structures [4]; Bluetooth-based MANET are small and quite static networks, also identified as Personal Area Networks (PAN), designed mainly to let printers and cell phones communicate when in direct and mutual visibility range [5]; IEEE 802.11-based MANET can consist of a large number of nodes, even geographically distributed, and generally widen to support multi-hop path routing [6].

MANET nodes are usually laptops, PDAs, or even smaller devices characterized by low computational capabilities and strict battery constraints. Their transmission technologies offer low bandwidth, high latency, high error rates, recurrent transmission collisions, and limited communication ranges (normally up to a few hundred meters). In addition, frequent node mobility causes weak link connectivity and possible temporary disconnections, even leading to network partitioning if some nodes leave their position of bridge between two network segments. All the above aspects affect significantly not only the crucial low-level issues of network auto-configuration and routing, as extensively described in Section II.A., but also the design and implementation of distributed applications for MANET, which is the main focus of the paper.

In fact, we claim that traditional solutions, at both the support level and the service one, are not suitable for the MANET environment, where peer-to-peer relationships are the only ones allowed according to the intrinsic nature of the network. Even if it were reasonable to assume the existence of a middleware/service backbone infrastructure in any MANET, the frequent mobility of the backbone nodes would make the overhead to repair the backbone often unsustainable. For instance, let us consider a support issue such as the setting of some network parameters (IP address, netmask and default gateway) at the MANET nodes. Obviously, no pre-configuration of nodes can be assumed, so any device should configure only when joining a network. In MANET it is unreasonable to rely on the presence of a centralized DHCP server because the server could become suddenly unreachable (with any provided service). Therefore, it is necessary an auto-configuration mechanism in charge of setting the parameters dynamically, by using a completely distributed protocol.

Other complex challenging issues have to be investigated to support service development and deployment in highly dynamic MANET environments. In MANET, terminal mobility is typically unconstrained, and hosts can freely move out of mutual coverage areas at any time, thus becoming suddenly unreachable, together with their possibly provided services and shared resources. In this context, even well established models, such as the traditional client-server one, require a significant re-thinking. The connection maintenance

becomes crucial because services could be unexpectedly interrupted due to server leaving or, symmetrically, to client movements. In addition, the frequent autonomous rebinding to services requires to face additional issues like the choice of the best server among a set of available ones, also depending on low-level connectivity conditions, e.g., the currently greater client/server signal strength. In such a scenario, the paper considers central to propose a middleware solution for guaranteeing service continuity independently of the mutual movements of clients and servers, which may loose direct single-hop connectivity at service provisioning time. In other words, we focus on flexible and highly dynamic middlewares capable of client/server rebinding and of continuing service delivery within a session in a completely seamless way, by minimizing the impact on the application logic and the complexity of developing clients/servers for MANET environments.

A. Related Work

MANET have recently stimulated several research activities, which are exploring different aspects of the challenging issues raised by this novel and highly dynamic mobile scenario. In these first years, the research work has mainly focused on investigating solutions, usually at the network layer, for the lower-level crucial problems imposed by MANET connectivity: mainly for the autonomous and decentralized configuration of MANET nodes and for the definition of highly dynamic multi-hop routing protocols. To the best of our knowledge, there are not yet application-level middleware solutions for MANET that address, as their main issue, the support and simplification of the development of client/server applications with service continuity, and our approach provides an original perspective in the field.

From the point of view of auto-configuration in networks without any infrastructure support, the main common goal of all the examined solutions is to automatically manage a temporary IP lease, i.e., to provide any terminal that joins a network with a valid and unique IP address, and to register the IPs returning free when the terminal exits the network locality, without the need of explicit operations by network administrators. Let us observe that well-established solutions for IP dynamic assignment in fixed networks, such as DHCP, require the availability of a deployed support infrastructure and do not fit highly mobile peer-to-peer MANET environments.

Four major projects have recently raised significant interest: on the one hand, Zeroconf [7] and IPv6 Stateless Address Auto-configuration [8] propose general solutions for traditional wired networks; on the other hand, PMWRS [9] and MANETconf [10] are designed specifically to work in MANET environments. The four auto-configuration solutions above address IP allocation problem in a quite similar way. In fact, they conceptually undertake the same sequence of actions: address generation, verification and final assignment. Therefore, at first, a tentative address is generated for the entering terminal. Two main classes of solutions are exploited: the extraction of the tentative IP from a pool known to all nodes participating in the network, or the construction of the tentative IP so as to be probably unique. In any case, the tentative IP may belong to another terminal and has to be verified. This is usually done by sending, according to each proposed solution, a packet with the tentative address to all nodes belonging to the network, so that they can verify not to own the same address (Duplicate Address Detection). If another host already owns the tentative address, it sends a negative reply to the sender, by forcing the generation of another tentative value. Reply messages exploit either broadcast communications (most common solution) or a direct dispatch, e.g., if the sender is known via a MAC address in local networks.

From the point of view of multi-hop routing protocols, several recent proposals have presented original solutions. If two MANET nodes, not in direct communication range, need to exchange messages, they have to rely on the forwarding ability of nodes located in the intermediate area. Several aspects make this forwarding problem hard to solve effectively in MANET: the dynamic topology makes most traditional routing algorithms not applicable; assigning the router role to any MANET node, although apparently natural in this context, may put an intolerable computational burden on unprepared devices; the wireless propagation implies undefined coverage areas that dynamically change, and the available bandwidth is limited, also because of possibly high error rates due to interferences [11]. In [12], different possible aspects to consider when providing taxonomy of MANET routing protocols have been identified: which routing information is exchanged, when and how this information is exchanged, when and how routing paths are built. Here we propose to combine principles suggested by different researches ([12], [13], [14]), to classify routing solutions in position-based ones, which exploit the knowledge about the geographic localization of the receiver, and topology-based ones, which consider the network as a link sequence. The latter can be split up in table-driven, on-demand and hybrid approaches.

Position-based protocols use a location service to determine the receiver position and so do not require route building and maintaining. Once achieved the visibility of MANET node location, the delivery of messages considers the mutual position of senders and receivers, and of the intermediate nodes physically located in the between. Three different forwarding strategies are generally employed: in greedy forwarding, packets are sent to a selected node on the receiver direction; in restricted directional flooding, to all nodes on the receiver direction; hierarchical approaches employ greedy forwarding for long ranges and topology-based algorithms for local routing [13].

Among topology-based solutions, traditional table-driven protocols (distance vector and link state families of algorithms) have demonstrated to be inefficient due to the low bandwidth and the frequent topology changes typical of MANET. On-demand protocols create a route only when

required by the source node. The communication requires a first phase in which the sender has to find a route to the destination; while topology conditions are considered unchanged, the calculated route is maintained valid. Hybrid protocols [15], instead, take advantage of traditional table-driven routing schemes into local communication contexts, and combine them with on-demand solutions for non-local routing.

III. The Design and Implementation of a Proxy-based Middleware for Service Continuity in MANET

Consider the usual scenario where a server, responsible for service delivery inside a MANET locality, i.e., the local PAN consisting of all the nodes in direct network visibility, suddenly and transparently leaves the locality during the service session. In this situation, the clients in the locality have to reorganize to reach the server independently of its movement, and to continue service session seamlessly. Depending on service implementation, the clients could either look for another equivalent server (if the provided service is stateless or the session state is maintained at the client side and exchanged at the server re-connection), or search for exactly the same server instance that left the locality (if the service is stateful and the state is exclusively stored at the server side). We claim the need of a middleware solution to support and facilitate the application development in this context, by permitting programmers not to bother about MANET node mobility and to focus only on the application-specific service logic. To this purpose, we have designed and implemented a middleware that takes care of location visibility and rebinding operations, thus allowing a simple, traditional and transparent client/server programming model. The middleware is in charge of control operations due to the established client/server relationships. It supports the discovery phase, by guaranteeing that clients could efficiently find the needed servers in the locality. When the servers possibly move, the middleware exploits client/server location visibility to reorganize the locality via the dynamic election of a proxy agent. The proxy takes care of searching servers by need, of forwarding client requests, and of performing multi-hop routing; it permits to organize solutions for client/server rebinding and service reestablishment that are scalable and mobility-transparent. In the remainder of the section, the paper presents how the middleware supports the different service phases sketched above.

First of all, the proposed middleware supports clients and servers during the setup phase of the service session. Clients can exploit middleware functions to perform a simple discovery operation, which consists in broadcasting a message to a service-specific port number and waiting for a suitable reply. Servers can invoke the middleware to exploit a simple lookup service that responds to client requests by disclosing the server identity, and to support the client/server connection (re-)establishment transparently to the application programmers.

When a server leaves a locality, the provided services would become immediately unavailable for all currently served clients. To enable service continuity, our middleware realizes search and restore phases transparently to clients and servers. First, it is necessary for the middleware to acquire visibility of the server movement at the client side. Two design choices are possible: the server (or a middleware component at the server side) could explicitly notify all interested clients of its mobility intention; or, clients could autonomously become aware of the server movement, transparently from the server-side point of view. The first choice can bring an excessive computational burden on the server: the server should maintain the list of the currently active sessions and of the correspondingly served clients in order to notify its movements to all interested entities; this notification may also be a heavyweight process when clients do not belong to the same locality of the server and require multi-hop routing solutions. Therefore, we decide to adopt the second design choice, by providing portable middleware functions to achieve client-side visibility of MANET node location. The middleware implements the location visibility functionality on top of the Linux Wireless Extensions and of the Microsoft Network Driver Interface Specification; an extensive description of the solution is out of the scope of the paper and can be found in [16].

However, if all interested clients try to locate a just moved server, certainly the hotspot of contemporary search messages can produce a local MANET congestion. In addition, not any client can own the needed implementation mechanisms to perform discovery operations, to choose among the set of retrieved equivalent servers, and to route service requests over multi-hop paths. To address all these issues, our middleware chooses an architectural solution based on the dynamic designation of a proxy for the locality. For instance, by considering the discovery operations, the middleware proxy is the only entity in charge of searching for the migrated server, thus achieving a scalable solution because the discovery traffic does not depend on the number of clients in the MANET locality. In addition, the proxy is the only software component that needs to know how to perform inter-locality routing operations.

Once the proxy finds the server, the proxy starts forwarding service requests/responses from/to interested clients. In other words, all service messages are automatically and transparently sent through the proxy, acting as a bridge between the clients and the server. Let us note that our proxy-based approach at the application-level permits also to enforce local management policies by adequately programming the support proxy, e.g., by restricting the number of contemporary active service sessions to impede an excessive degradation of the locally available bandwidth.

The introduction of a support proxy in a MANET locality, where we cannot assume the availability of any static infrastructure, is possible only if the proxy is conceived as a totally dynamic role, assigned to one of the local clients in a completely distributed and decentralized way. Our distributed

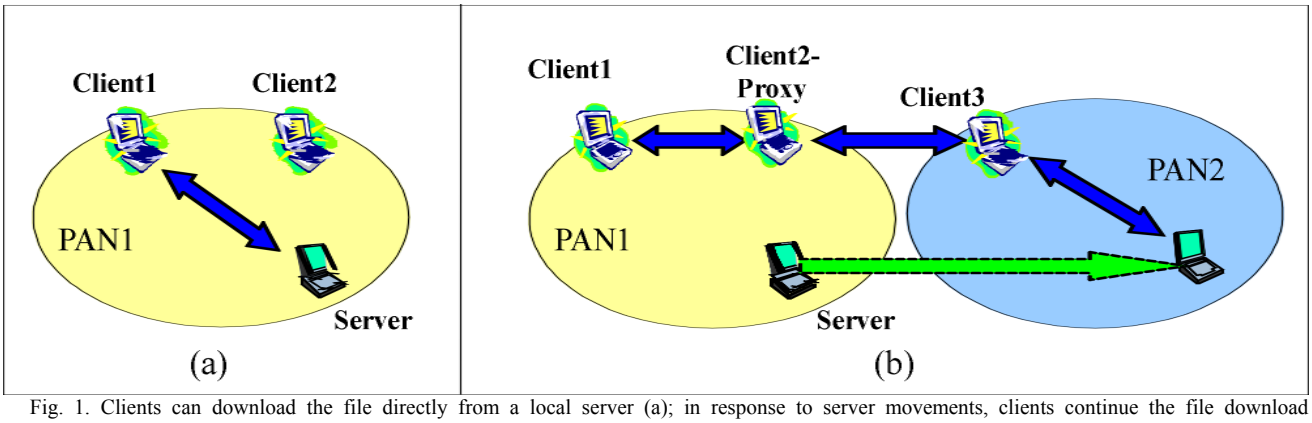

Fig. 1. Clients can download the file directly from a local server (a); in response to server movements, clients continue the file download transparently via the locally elected proxy (b).

middleware assigns the proxy role via an ad hoc lightweight election protocol, described in the following. In addition, once elected, if the client does not host the behavior necessary to act as a proxy, it can download the needed code from discoverable and disseminated code repositories, realized to support the SOMA mobile agent platform [1], or from a local client peer. This dynamic code download increases the middleware dynamicity and enables the runtime deployment of new implementations without imposing service interruptions.

The leader election problem and the specification of distributed election protocols have been extensively investigated in wired networks with static topology. The MANET mobility significantly changes the perspective and makes most election protocols unsuitable. Frequent link/node failures induce disconnections, which in turn could lead to network partitioning, with the consequent issues stemming from elections in disjointed segments and conflict resolution protocols in case of locality merging. Other MANET-specific issues are due to high error rates and possibly long delays in message delivery. For these reasons, we decided to adopt a novel and MANET-fitting election solution, with the main goal of maintaining the protocol very simple and lightweight, by considering the usual strict constraints on resource availability over MANET devices. We implemented a variant of the "bully algorithm". During the discovery phase, the server replies to clients not only disclosing its identity, but also assigning to each client a unique identifier, based on client characteristics sent within the service request message. For instance, the faster is the wireless connectivity of the client, the greater the assigned identifier. The election protocol is then triggered when a client senses the server movement: the client immediately broadcasts its identifier; when receiving this message, any client compares its identifier with the received one, and broadcasts a reply message with its own identifier if and only if the latter is greater than the received one. The only node that does not receive replies within a timeout supposes to be the elected proxy. The implemented election protocol also considers message losses, and provides a series of countermeasures to guarantee the election consistency in a wide set of temporary failure cases. Details of the election protocol are out of the scope of the paper, which in the following exemplifies how the proxy-based middleware simplifies the development of MANET applications with service continuity.

IV. The File Transfer in Mobile Ad-hoc Networks (FT-MAN) Application

Several different types of applications can be built on top of the proposed general-purpose middleware for MANET. In particular, we have implemented a prototype of remote file transfer service over our MANET support to measure the middleware performance in a practical usage scenario and to show how the design and implementation of applications turns out extremely simplified. FT-MAN provides file downloading from a dynamically discovered service component available in a MANET locality, even if the server moves during file transfer. In the case of a server exiting from the locality of its clients, the middleware achieves visibility of the server movement and triggers the transparent client reorganization; from the client point of view, file transfer operations that were active at the moment of server migration and new file transfer requests continue to go on seamlessly. As we detailed in the previous section, the election of a proxy occurs in the old server locality. After the election, the proxy substitutes the server in the locality, and the clients can continue to query the proxy exactly as it were the local server. Actually, communications go from the clients to their local proxy, and from the proxy to the retrieved server; file transfer replies run along the reverse path, as depicted in Figure 1. In this way, the proxy supports the multi-hop routing of service packets: the proxy acquires routing information during the server discovery phase and adds this routing data to the service packet header. Routing algorithms could be also changed during service provisioning, since the middleware can provide the proxy with new implementation code dynamically downloaded from distributed code repositories, which can be discovered at runtime uniformly as the other services, e.g., FT-MAN servers.

In addition, at any transparent re-establishment of the client/server connection via the intermediary proxy, FT-MAN

clients and servers perform a state information exchange protocol. Figure 2 shows that, at first, the client sends a message to the server by indicating the name of the requested file and how many bites have already been downloaded. In the reply, the server states that either the file is unavailable, e.g., in the case of wrong filename, or the transmission can start. In the last case, a server message with the file dimension is sent to the client before starting the content delivery. It is the middleware that triggers the clients to exchange the current session state when needed; the client is only required to implement a well-known method for state transmission.

In the FT-MAN prototype we have decided to maintain the session state on the client side to avoid overloading the server, which is usually in charge of generating most file transfer traffic. In particular, the client maintains the number of bytes correctly received (by saving this information every 10KB transferred), in order to allow the download resuming when the direct client/server connection stops. As stated in the previous section, since the state is stored on the client side and is exchanged at the beginning of the application protocol, the proxy can also decide a re-connection with an equivalent server, without the constraint of looking for only the server instance originally present in the client locality.

The implementation of the FT-MAN client and server on top of the proposed middleware is really simple. Only two classes are needed: a module that carries on the server side of the protocol, and a client component that reads the transferred bytes from the input stream and stores them locally. The implementation classes do not bother of mobility issues, but concentrate only on the application logic, with minimal differences with regards to the implementation of the same service in wired and static deployment scenarios.

V. Experimental Evaluations and Performance Results

To measure the middleware performance and to quantitatively verify the feasibility of our approach, we have deployed the middleware and FT-MAN on a little testbed consisting of a couple of MANET localities with a few nodes (IEEE 802.11b-compliant portable devices that exploit the WiFi ad hoc connectivity mode). To better evaluate the behavior of the system under actual operating conditions, we have deployed it over two different types of devices, Acer laptops and Compaq-HP iPAQ PDAs (with the PC CARD Expansion Set), both outfitted with Cisco 350 Client Adapters. The former run the Linux operating system (RedHat 8.0 with 2.4.20 custom kernel), the latter use the Familiar Linux operating system, version 0.6. The multi-hop message delivery was obtained by employing a routing protocol, AODV-UU, implemented on the Linux kernel.

Our middleware is implemented by using the Java programming language because the running environment provided by the Java Virtual Machine (JVM) offers a transparent interface with system level services and functions. In the evaluation testbed we have installed the J2SE 1.4.0 on the laptops and the Blackdown JVM 1.3.1 implementation on the PDAs.

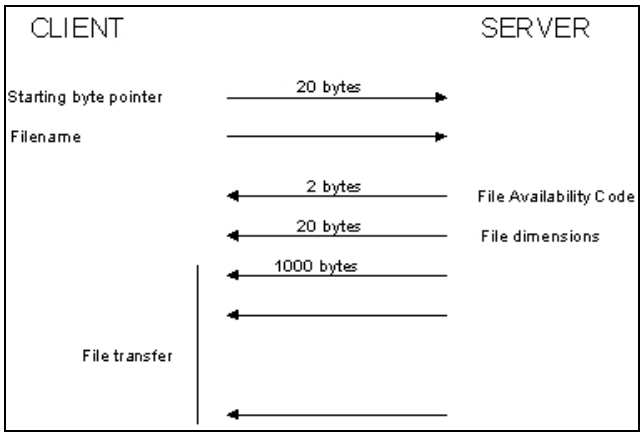

Fig. 2. The client/server exchange protocol..

At first, we measured how long it takes for an FT-MAN client to complete the transfer of a 10 MB file, directly from a server that does not move during service provisioning (Table 1). In such a situation the middleware do not get involved, since the entities remain in mutual direct visibility. Then, we accomplished several tests by changing the value of two main middleware configuration parameters: *IAPTime*, the waiting time before accepting the result of proxy election (to simulate networks of increasing dimensions) and *SearchPeriod*, the interval between successive server searches (to vary the proxy reaction promptness and to test in which conditions the control messages flood the MANET locality). Finally, we organized tests while changing the *inactivity interval*, defined as the period of server unavailability due to its change of locality; *inactivity interval* depends on server mobility characteristics and on the time needed to restart the service when entered in a new locality.

IAPTime and *SearchPeriod* significantly impact on the overall middleware performance and should be tuned carefully depending on the specific deployment scenario and on application-specific requirements. *IAPTime* influences the election protocol completion time: it should depend on the average number of clients supposed in the locality because, when *IAPTime* expires, all participant clients should have had the chance to broadcast their identifiers. A low value of *SearchPeriod*, instead, lets the proxy determine promptly the new server location, but increases the control message overhead if the server remains disconnected/unreachable for a long time and no equivalent service components are discovered. To obtain a flexible tuning and to avoid code recompilations, we decided to take out the principal middleware configuration parameters from the code and to put them in XML files examined at the application launch. The configuration files used in FT-MAN are structured as in the example in Figure 3.

We have measured the average values of download time, service unavailability at client, and search period at proxy over a large set of experiments. Table 2 presents the results when a server component leaves the MANET locality by

```
<Config:ConfList
xmlns:Config="http://www.deis.unibo.it">
<Config:Param>
      <Config:Type>IAPTime</Config:Type>
      <Config:Value>200</Config:Value>
</Config:Param>
…
</Config:ConfList>
```

Fig. 3. The XML configuration file format.

remaining active, and reconnects via the proxy when entered in a new locality. An *inactivity interval* of 400ms has been fictitiously introduced to model the time needed to the interested clients to poll the information about the server leaving. *IAPTime* and *SearchPeriod* were fixed to a low value of 200ms: this *IAPTime* interval has emerged as the minimum necessary in our testbed to avoid election problems, i.e., clients that erroneously believe to be the proxy not having received control messages in time.

A correct tuning of *IAPTime* and *SearchPeriod* permits to minimize the time required to reconnect the server. On the one hand, if *inactivity interval* is low, it could be worth to tune also *IAPTime* and *SearchPeriod* to low values, by paying a greater overhead but obtaining a more reactive system. On the other hand, in several application deployment scenarios there are no strict promptness requirements, especially in highly dynamic and mobile environments: *inactivity interval* is high and, therefore, both *IAPTime* and *SearchPeriod* can be tuned to high values. We have designed the middleware to be self-adaptive: *IAPTime* and *SearchPeriod* start with common intermediate values and then are modified depending on monitoring data about the service unavailability history and the average number of local clients. Table 3 reports the performance results when the server *inactivity interval* is 5s. By taking into account the previous considerations, in this case *IAPTime* and *SearchPeriod* automatically set to values in the same range of *inactivity interval* (1s in the table).

The reported results show that the overhead due to control message transmission (including election and server search phases) can be always quantified in about 200ms. It is encouraging that most of the delay can be ascribed to the setting of the different configuration parameters.

Parameter	Average Value (ms)	Std. Dev. (ms)
Download Time	28154.4	428.04

Table 1. Download of a 10 MB file directly from server.

Parameter	Average Value (ms)	Std. Dev. (ms)
Service Unavailability on Client	658.6	11.01
Search Time on Proxy	448.4	9.18
Download Time	35502.7	441.01

Table 2. Test conditions: server inactivity interval = 400ms; IAPTime = 200ms; SearchPeriod = 200ms.

Parameters	Average Value (ms)	Std. Dev. (ms)
Service Unavailability on Client	5223.4	23.10
Search Time on Proxy	5052.6	33.68
Download Time	40232.6	450.12

Table 3. Test conditions: server inactivity interval = 5s; IAPTime = 1s; SearchPeriod = 1s.

VI. Conclusions and On-going Work

The development and deployment of distributed applications in MANET require flexible and mobile middleware solutions capable of properly handling the frequent changes in local device/service visibility during service provisioning. In addition, the complexity imposed by the MANET scenario motivates a clear separation of concerns between the client/server application logic and the support solutions to discover, rebind, and route requests to mobile service components. Novel middleware solutions based on code mobility can effectively provide this separation of concerns and achieve the level of dynamicity, flexibility and reusability requested in highly dynamic network environments.

First experiences coming from the prototype deployment and testing have shown that the proposed middleware can significantly facilitate the design and implementation of MANET applications with feasible performance results, thus potentially leveraging the promising market of PAN services. These encouraging results are stimulating further research to extend the framework in two main directions to offer a widespread support for mobility in MANET. First, we are working on an exhaustive integration of our middleware prototype with all the multi-hop routing protocols available in the literature, and on the definition of election protocols capable of clustering clients depending on specified metrics, e.g., based on received signal strength. In addition, we are exploring solutions to dynamically establish proxy-to-proxy inter-locality chains; this can speed up and facilitate the search of non-local suitable servers and can significantly improve the scalability of the proposed middleware.

References

[1] P. Bellavista, A. Corradi, R. Montanari, C. Stefanelli, “Dynamic binding in mobile applications: a middleware approach”, *IEEE Internet Computing*, Vol. 7, No. 2, pages 34-42, Mar.-Apr. 2003.

[2] J. Macker, S. Corson, “Mobile Ad-hoc Networks (MANET)”, http://www.ietf.org/ html.charters/manet-charter.html, 1997.

[3] J. Bolliger, T. Gross, “A framework-based approach to the development of network-aware applications”, *IEEE Transactions on Software Engineering*, Vol. 24, No. 5, May 1998, pp. 376-90.

[4] L. Clare, G. Pottie, J. Agre, "Self-organizing distributed sensor networks", SPIE Conf. Unattended Ground Sensor Technologies and Applications, pp.229-237, 1999.

[5] Bluetooth SIG Inc. - *Bluetooth*, http://www.bluetooth.com

[6] IEEE 802.11b Working Group, "Wireless LAN medium access control (MAC) and physical layer (PHY) specifications: higher speed physical layer (PHY) extension in the 2.4 GHz band", http://grouper.ieee.org/groups/802/11/, 1999.

[7] S. Cheshire, B. Aboda, E. Guttman, "Dynamic configuration of IPv4 link-local addresses", IETF Memo, http://files.zeroconf.org/draft-ietf-zeroconf-ipv4-linklocal.txt, Aug. 2002.

[8] S. Thomson, T. Narten, "IPv6 stateless address autoconfiguration", IETF RFC 2462, http://www.ietf.org/rfc/rfc2462.txt. Dec. 1998.

[9] C. Perkins, J. T. Malinen, R. Wakikawa, E. Royer, Y. Sun, "IP address autoconfiguration for Ad Hoc networks", IETF MANET Working Group Internet Draft, http://www.ietf.org/ID.html, Nov. 2001.

[10] S. Nesargi, R. Prakash, "MANETconf: configuration of hosts in a mobile Ad Hoc network", IEEE INFOCOM, pages. 1059-1068, 2002.

[11] D. B. Johnson, "Routing in Ad Hoc networks of mobile hosts", IEEE Workshop on Mobile Computing Systems and Applications, Dec. 1994.

[12] X. Zou, B. Ramamurthy, S. Magliveras, "Routing techniques in wireless Ad Hoc networks", 6th World Multiconference on Systemics, Cybernetics, and Informatics, July 2000.

[13] M. Mauve, J. Widmer, H. Hartenstein, "A survey on position-based routing in mobile Ad Hoc networks", *IEEE Network*, Nov./Dec. 2001.

[14] E. M. Royer, C.-K. Toh, "A review of current routing protocols for Ad Hoc mobile wireless networks", *IEEE Personal Communications*, Apr. 1999.

[15] J. C. Requena, N. Beijar, R. Kantola, "Replication of routing tables for mobility management in Ad Hoc networks", *ACM Wireless Networks (WINET) Journal*, 2003.

[16] P. Bellavista, A. Corradi, "Mobile middleware solutions for the adaptive management of multimedia QoS to wireless portable devices", to be presented at IEEE Int. Workshop on Object-oriented Real-time Dependable Systems (WORDS), Naples, Oct. 2003.