Lightweight Code Mobility for Proxy-based Service Rebinding in MANET

Paolo Bellavista, Antonio Corradi, Eugenio Magistretti
Dip. Elettronica, Informatica e Sistemistica - Università di Bologna
Viale Risorgimento, 2 - 40136 Bologna - ITALY
{pbellavista, acorradi, emagistretti}@deis.unibo.it

Abstract— **Advances in device miniaturization and wireless technologies are stimulating Mobile Ad-hoc NETworks (MANET) where mobile nodes can autonomously organize in a peer-to-peer mode, without requiring a statically deployed network infrastructure. Because of node mobility, the set of service components that MANET clients have at one-hop distance changes often during service provisioning. That continuous change in locally accessible services significantly increases the complexity of designing and implementing effective MANET applications. The paper claims the need of dynamic middleware supports for MANET, not only to address routing/configuration issues, but also to automate the re-qualification of service bindings at provision time. It presents COMMAND, a flexible middleware solution, based on code mobility, for transparent service rebinding in MANET. COMMAND exploits dynamically elected proxies that act as intermediaries to decouple mobile clients and needed service components. In particular, the paper focuses on how COMMAND implements a lightweight MANET-specific solution for effective code distribution to deploy the needed proxy behavior only when and where required. Finally, the paper shows the implementation of a COMMAND-based forum distributed application, together with its related performance, to point out how the proposed middleware can help service development in MANET.**

Keywords: MANET; Middleware; Service Reconfiguration; Code Mobility; Proxy

I. INTRODUCTION

The necessity of rapid, flexible and temporary connections between possibly heterogeneous mobile devices has recently motivated intense research activities in the Mobile Ad-hoc NETworks (MANET) area [1]. MANET nodes can move at any time, even during service provisioning, while a client node has started to access distributed server components in the MANET but not terminated yet. Node mobility and consequent variations in network topology force continuous network reorganizations. Due to the temporary and spontaneous nature of MANET connections, it is almost impossible to rely on a statically deployed network support infrastructure. MANET nodes tend to be autonomous entities that cannot guarantee a durable and continuous presence in cooperating and performing multi-hop information delivery [2].

The high dynamicity of MANET makes the design and implementation of distributed applications significantly more complex than in traditional wired environments. In particular, most development challenges stem from two key MANET properties: lack of a support infrastructure and high mobility of terminals. On the one hand, several infrastructure-based solutions, effective in wired networks, hardly suit MANET environments. For instance, in MANET it is unlikely to assume that a configuration server is continuously available to provide the needed network configuration data, such as temporarily assigned IP addresses in DHCP. As a consequence, MANET require the design and implementation of completely distributed ad-hoc protocols for dynamic host configuration. On the other hand, node mobility may cause frequent disconnections of MANET clients and needed distributed resources, e.g., due to the loss of direct connectivity when either resources or clients move out of the reciprocal wireless coverage area and none can perform multi-hop routing.

MANET force to reconsider even well established distributed interaction models, such as the client/server one. For instance, clients cannot assume that, once discovered and bound to a suitable server component, their established connections could persist for the whole service session. In other words, the application logic should continuously verify and update the list of reachable service components, and manage the possible disconnections by performing rebinding operations accordingly. Let us note that, even when mutual node movements do not cause interruption of established connections, it could be relevant to dynamically re-qualify the bindings to service components to favor optimal exploitation of local (at single-hop distance) resources. For instance, when two reachable and functionally equivalent servers are visible, it is preferable to rebind to the currently local one in place of the other at multiple-hop distance from the client.

All the above motivations have led us to design and implement a highly dynamic and flexible middleware, called **CO**de Mobility **M**iddleware for **MAN**ET **D**ynamic Rebinding (COMMAND), with the main goal of supporting automatic service rebinding in MANET. COMMAND not only provides MANET nodes with configuration support, but also facilitates the development of MANET applications by automatically managing connection rebinding when clients and servers lose direct visibility, i.e., when they become more than one-hop distant, during service provisioning. The ultimate goal is to allow application developers to concentrate only on application logic and to design MANET distributed services as in wired de-

ployment scenarios with stable client/server connections. In addition, COMMAND decides to operate at the application level to simplify the achievement of high flexibility and full portability over different MANET implementations.

COMMAND is based on the primary idea of middleware proxies that act as decoupling components between client and server endpoints to support their binding/rebinding independently of mutual movements during service delivery. Due to intrinsic lack of infrastructure and high dynamicity of MANET, the choice of which nodes act as proxies is completely decentralized via an ad-hoc election protocol. The assumption of static availability of all needed middleware proxy components at all nodes, e.g., distributed search module or caching/filtering/transcoding functions in advanced service scenarios, is typically unfeasible for resource-constrained MANET devices. Thus, we claim that it is crucial to have MANET middleware supports capable of distributing the needed code at runtime, in a completely decentralized way, by considering the peculiarities of the MANET environments to optimize the adopted mobile code mechanisms. The paper focuses on a lightweight MANET-specific solution for code distribution to deploy the needed proxy behavior only when and where required at service provision time.

The paper is structured as follows. Section 2 presents the design guidelines and the architecture of our middleware for automatic service rebinding in MANET, while Section 3 is devoted specifically on the description of the COMMAND Code Mobility facility. Section 4 reports the experience made with the implementation of a COMMAND-based forum case study, which exemplifies how our middleware significantly facilitates the development of MANET applications. Section 5 reports related performance results. Conclusions and on-going research activities end the paper.

II. COMMAND: A PROXY-BASED MIDDLEWARE FOR AUTOMATIC REBINDING

MANET enable highly dynamic service scenarios characterized by the potential mobility of all participants. This challenging environment suggests to identify localities consisting of MANET nodes in direct wireless visibility and to provide distributed services by favoring local interaction. For the sake of presentation, let us consider the case of a server that belongs to one MANET locality and moves to another locality while running active service sessions. The server change of locality would either require the dynamic organization of routing chains of forwarding nodes to maintain the client-server connectivity or produce the abrupt interruption of the service sessions if MANET nodes in the locality do not support multiple-hop routing. To enable session continuity notwithstanding server movements, any client should be capable of reacting to server disconnection, understanding whether it is possible to rebind to an equivalent server in its locality, and performing a multi-hop inter-locality search for (and routing to) the moved server. Let us note that similar considerations apply to the case of client movements with regards to their needed and unmoved server, and to the combination of both movements.

We claim unviable a solution where any MANET client node should own all the above capabilities. First of all, MANET nodes are heterogeneous and often very resource-constrained: it is impossible to statically equip any node with all the functions possibly needed at runtime to allow session continuity. Secondly, charging application developers with the burden of implementing the session continuity support significantly complicates the realization of MANET applications, thus slowing down the emergence of this novel service market. Moreover, in the usual case of a high client/server ratio, the concurrent search for the moved server by several MANET clients in a locality is likely to produce local network congestion, by degrading other service sessions active in the same locality.

The above comments have motivated the design and implementation of the COMMAND middleware for transparent service rebinding in MANET. COMMAND automatically performs the needed rebinding in response to the change of locality of clients and servers when their movements cause the loss of their direct visibility, without the need of having multiple-hop routing solutions statically pre-installed on any MANET node. COMMAND exploits code mobility (as better detailed in the following), has been implemented in Java, and works over MANET nodes hosting the Java 2 Micro/Standard Edition with either IEEE 802.11b (in the ad-hoc configuration mode) or Bluetooth. COMMAND operates at the application level to facilitate its portability over different lower-level MANET solutions, e.g., over different node platforms and heterogeneous wireless connectivity technologies. The application-level choice is recognized suitable to provide flexible solutions to crucial mobility issues, such as application-specific information dissemination and caching, security, and interoperability because middleware supports can benefit from all the standard mechanisms, solutions and tools available at this abstraction layer [3].

The primary guideline of COMMAND is the introduction of middleware ***proxies*** to act as decoupling components between client and server endpoints. Proxies are in charge of performing the management operations to seamlessly rebind moved clients and servers, and of working as inter-locality forwarders for service requests/replies. One COMMAND proxy executes in each MANET locality, serves all local clients and maintains client/server connections transparently and persistently available for all the duration of the service session.

The proxy adoption can induce several advantages. First, only the MANET nodes hosting proxies have to own the know-how needed to find entities outside their locality and to re-establish the client/server sessions. This differentiation of node roles meets the heterogeneous (and often very limited) capabilities typical of MANET hosts. Secondly, the adoption of a locality proxy can facilitate the enforcement of local management policies, e.g., to restrict the maximum number of concurrent inter-locality active sessions in order to prevent an excessive degradation of the locally available bandwidth. Finally, application-level proxies could also cache service results and directly reply to local clients instead of remote servers, by reducing the need for non-local communications.

The introduction of support proxies in MANET localities, where a static infrastructure is not available, is possible only if proxies play a totally dynamic role, usually assigned to one of the local clients in a completely distributed and decentralized

way. COMMAND dynamically assigns the proxy role via an ad-hoc lightweight election protocol (see Section 2.B). Any node, since in principle it may become proxy, should be deployed with all COMMAND functions required for proxy. Because this assumption is unfeasible for resource-constrained MANET devices, we claim the importance of adopting ***code mobility*** mechanisms in middleware solutions as in COMMAND. By exploiting code mobility, any MANET node can retrieve the needed code at runtime. In particular, for MANET it is crucial to provide novel locality-aware solutions to distribute code that exploit intra-locality efficient communications and reduce the network traffic generated for code retrieval. The original COMMAND solution for code mobility is extensively described in Section 3.

A. Overview of the COMMAND Middleware Components

COMMAND offers a rich API to simplify the design of client/server application components for MANET. In particular, it provides a dynamically elected proxy component in any MANET locality. In addition, it offers client/server stubs for node configuration, local discovery, and message forwarding, as shown in Fig. 1. The Configuration facility permits clients/servers to join a COMMAND locality by providing them with unique identifiers (required for the proxy election protocol as discussed in Section 2.B) and initialization parameters. The client/server Discovery facility is in charge of broadcasting/responding to lookup service requests in the locality. The Data Send and Receive facility redirects requests/responses either to the client/server (when they are co-located) or to the intermediate proxy.

In the following, the paper focuses on the primary COMMAND component (the proxy) and on how it is possible to extend dynamically its behavior via the Code Mobility facility.

B. The COMMAND Proxy: Functions, Design, and Implementation

As already stated, when a server ruling active service sessions exits a locality, the COMMAND middleware triggers a distributed election protocol to choose the local node most suitable to host the proxy. The election is triggered if there is no proxy in the locality, and also whenever the current proxy leaves the locality before the termination of all active sessions.

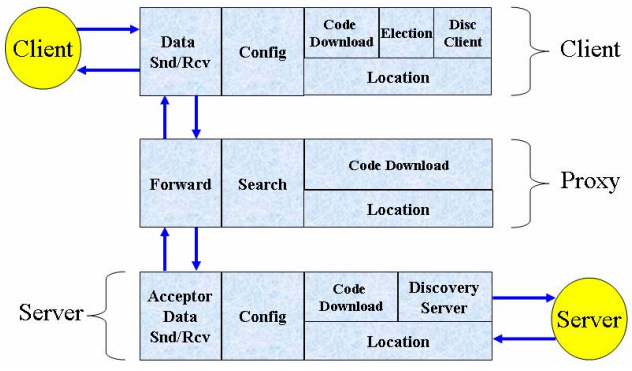

Figure 1. The COMMAND architecture: middleware components and data flows.

The election protocol estimates the suitability of each node of becoming proxy by considering its capabilities, e.g., computational power and available memory, and its expected mobility patterns based on its history of exhibited movement habits.

Frequent link/node failures, high error rates and possibly long delays in packet delivery make most traditional election protocol unsuitable for MANET. For these reasons, COMMAND adopts 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 by disclosing its identity, but also by 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 via the Location facility depicted in Fig. 1. In that case, 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.

The elected proxy is in charge of inter-locality server search and of client session re-establishment. To this purpose, the proxy exploits three main middleware facilities:

- the **Search** facility to retrieve the moved server component (or one functionally equivalent replica) outside the original locality;
- the **Forwarding** facility to transparently redirect client requests and server responses;
- the **Code Mobility** facility to dynamically retrieve the needed behavior after the election and when novel middleware components should be deployed.

Depending on the nature of the provided application-level service, the Search facility can look either for the same server instance moved out of the proxy locality (in the case of stateful services and when the session state is exclusively kept at the server side) or for an equivalent one (in the case of stateless services and when the session state could be maintained at the client side and delivered to the new server during the first service connection phase). To scan localities that are outside its direct wireless visibility, the proxy exploits the AODV multihop routing protocol [4]. Again, the code implementing the protocol, if not already present in the elected node, is dynamically downloaded when needed.

The Forwarding facility is exploited only after finding the server outside the proxy locality. This facility re-qualifies connections between clients and servers by acting as an active bridge. It does not blindly redirect client messages to the server (and vice versa), but inspects message content and decides the

actions to perform depending on current conditions in the locality. For instance, in the case of caching-enabled forwarding, the facility avoids to contact the server and directly send service results to the client, when possible, by querying locally cached previous results. Or, if network traffic in the locality is too high, it enqueues client messages while waiting for a decongested situation. Any client-to-server forwarding channel is handled by a dedicated thread, instantiated only when the proxy receives a new client request, without the static allocation of a thread pool, to reduce static computational load at the proxy.

III. THE CODE MOBILITY FACILITY

In dynamic and heterogeneous scenarios two primary considerations call for a mobile code facility. On the one hand, MANET devices are generally provided with scarce amounts of memory, thus one cannot expect that each peer owns all code to execute all possible tasks. This holds especially for proxy peer duties. On the other hand, we claim the need of improving the flexibility of the middleware by supplying a mechanism to update component behavior at runtime. This mechanism enables the installation of facilities when needed and their discarding after service provisioning.

Even if the mobile code research have achieved relevant results in the last years, especially when supporting mobile computing in both traditional wired networks and infrastructure-based wireless environments [5-8], no appropriate solutions for code mobility in MANET have been proposed yet. In fact, it is crucial that MANET-specific code mobility considers two primary points deriving from the peculiar deployment environment:

- inter-locality communication is much more expensive than in wired LANs or cellular wireless networks, because MANET require a significant resource consumption by the forwarding nodes along the sender-receiver paths;
- code transfers (as any other communication) are power-consuming operations. Thus, mobile code mechanisms should be designed to minimize energy consumption and performed only when strictly needed.

These considerations have motivated the two main guidelines of the COMMAND code mobility support:

- the COMMAND code transfer exploits the code already available within the locality as much as possible, by looking for non-local code repositories only when no closer copies are available;
- COMMAND adopts metacode to spread, in a concise and effective way, the knowledge about all code modules (and versions) currently available in the MANET locality.

In more details, COMMAND exploits metacode descriptors to maintain information about the code available on all MANET nodes in the locality. Each descriptor specifies name and version of stored code modules through a sequence of XML-based elements [9]. In particular, each COMMAND node maintains two repositories: the Local Code Repository with the descriptors and the code of the modules installed on the node; the Neighbor Metacode Repository describing the code modules available at neighbor nodes, together with their location.

Every time a COMMAND node needs a code module, it searches the metacode information stored in its local repository. For instance, when a node is elected proxy (or when it requires an updated version of the election protocol), it first checks whether its Neighbor Metacode Repository includes a descriptor for the required module. If the descriptor is found, the node downloads the code directly from the referred local node. Anytime a node downloads a code module, the Local Code Repository is updated. When the repository reaches its capacity limits, COMMAND discards entries by following a least recently used replacement policy. If no suitable entry is found (or if the node registered for that code is no longer available in the locality), the requesting node asks its COMMAND proxy to perform a search outside the locality on its behalf.

Let us rapidly observe that, thanks to the COMMAND support, a code-requesting node requires being capable of performing multiple-hop routing only in the case that it is looking for the code to act as a newly elected proxy and that there is no local node already owning that code.

In other words, the COMMAND code mobility protocol follows these default rules:

1. a piece of code is downloaded only when strictly necessary, i.e., when the Java class-loader cannot find locally the needed packages at execution time;
2. a node entering a new locality advertises its code, by single-hop broadcasting the content of its Local Code Repository;
3. when a node leaves a locality, COMMAND notifies the event to all other local nodes for their autonomous updating of Neighbor Metacode Repositories;
4. when a node requires and obtains a piece of code available inside the locality, all local nodes can decide to download the code at the same time (depending on their available memory). That exploits the broadcast nature of the local wireless communications;
5. if the required piece of code is not available in the locality, the requesting node asks the proxy for a remote search. As soon as the code is found, the proxy automatically broadcasts it in the locality. Local nodes may update their Local Code Repository similarly to the case above.

These default rules can be modified by introducing suitable policies written in the Ponder language [10]. Since policies can be dynamically modified, the behavior of the Code Mobility facility can be changed at runtime, without affecting its implementation code. The architecture of this facility is shown in Fig. 2. The Download Mechanism is responsible for code/metacode sending/receiving and for the updating of the corresponding repositories. The Policy Manager reacts to the events arisen by the Monitoring component by calling the Download Mechanism according to the specified policies. A thorough description of the Monitoring component and of the

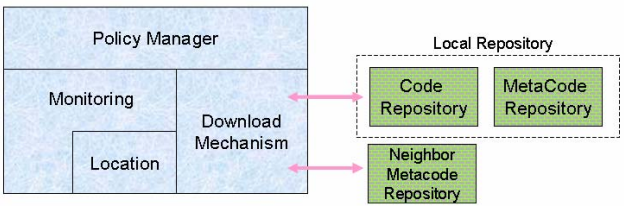

Figure 2. COMMAND message sequence during service rebinding in Forum-MAN

Policy Manager is out of the scope of the paper; additional details are in [11].

Let us finally explain, with an example, how the Code Mobility facility behavior can change. Every time a peer enters a locality, the Monitoring component senses its joining and raises an event. This event triggers the associated policy (described at point 1) that obliges the node entering a locality to broadcast the content of its Local Metacode Repository. If a MANET administrator prefers to save the bandwidth at the entrance of a new peer in the locality and, at the same time, is willing to accept a temporary lack of consistency in the state of its participant repositories, she needs only to define a new policy specification with no action associated to node entrance. This decision and the different policy adopted has absolutely no impact on the implementation of the Code Mobility facility.

IV. THE FORUM-MAN CASE STUDY

COMMAND is a general-purpose middleware for MANET and different types of applications can be built on top of it. To verify the behavior of the COMMAND middleware in a practical usage scenario and to show how it facilitates the design and implementation of applications, we have developed the Forum-MAN (Forum in Mobile Ad-hoc Networks) application prototype. Forum-MAN is organized in rooms (thematic channels). Participants can post and get messages to/from their preferred channels. Once posted, a message can be modified only by authors or by channel administrators. This service is built to promote direct interactions between close neighbors, and also to maintain the possibility to continue participating in interesting channels in case of mutual movements of mobile clients and servers. In fact, Forum-MAN provides users the possibility to dynamically rebind to server components and to use services even after server movements.

Let us explain how the application works on top of COMMAND. At first, each client joining a new locality connects to the server and accesses services by employing the traditional send and receive primitives provided by COMMAND. Behind the primitives, the middleware executes a transparent discovery of the server peer and provides application-level message delivery. The server offers Forum-MAN services inside its original locality by publishing messages posted on managed channels. Hence, local peers can send their updates and messages directly to the server. When the server decides to leave the locality, clients can continue to use the service transparently. In fact, an elected proxy undertakes the server search operations and establishes connection rebinding. As soon as it finds the server, the proxy notifies clients of the Forum-MAN service re-establishment within the locality. Therefore, the middleware components running on clients connect to the Forwarding facility of the proxy, which provides for the relaying of requests to the moved away Forum-MAN server (Fig. 3). The Forwarding facility behaves like a multi-hop routing functionality: it acquires routing information during the server remote search phase and adds this information to each service packet header.

In addition, we claim that the Code Mobility facility improves application flexibility. For instance, routing modules unavailable at proxy and new releases of the Forum-MAN client can be downloaded during service delivery. COMMAND provides the dynamic deployment of new code delivered on peers by downloading from code repositories, to be discovered at runtime similarly to (and by using the same support as) all other application services. COMMAND performs all code management operations without any explicit intervention of the application components (and hence without any burden for the application client/server developer).

The implementation of the Forum-MAN client and server on top of the proposed middleware is very simple. The client component, at first, exploits the COMMAND Discovery to find which peers run the server implementation. Then, it sends to the server the messages inserted in the Forum-MAN GUI by the local user and shows notes posted by other users, by employing the communication primitives provided by the Data Send and Receive facility. The client components also maintain the state related to the messages already obtained from the server. By attaching this state as a parameter to each updating request, clients can receive packets containing only recently posted messages. Likewise, the server component at first registers a new entry in the Discovery Server of its local peer. Afterwards, every time the Data Send and Receive facility running on that peer receives a new request, it invokes a specified method of the server. This method, depending on request type, publishes new messages posted by the users or propagates messages recently posted to clients. The implementation classes are completely unaware of all mobility issues, but concentrate only on the application logic, with minimal differences if compared with the implementation of the same service in wired and static deployment scenarios.

V. EXPERIMENTAL EVALUATIONS AND PERFORMANCE RESULTS

To quantitatively verify the feasibility of the approach based on dynamically elected proxies, we deployed the COMMAND-based Forum-MAN application in a little testbed

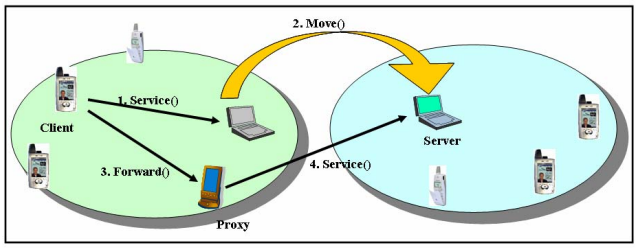

Figure 3. COMMAND message sequence during service rebinding in Forum-MAN

within our Department Wireless Lab. We organized our devices (Acer TM518 laptops running the Linux 2.4.20 kernel and Compaq iPAQ PDAs running the Familiar Linux distribution) in two heterogeneous and disjoint MANET localities, as depicted in Fig. 3. The communication was enabled by IEEE 802.11b-compliant devices (Cisco 350 Client Adapters) configured in ad-hoc mode.

We have evaluated the Forum-MAN performance when the server moves from one locality to the other. We have decided to measure two primary time indicators: *Service Unavailability on Client* represents the time interval between client/server connection loss and re-establishment; *Search Time on Proxy* is the interval for the proxy to find the moved server. We have measured average values of the two above indicators over a large set of experiments. The obtained results point out that the performance of the COMMAND middleware is definitely compatible with the time constraints of most classes of MANET applications with no strict real-time requirements on service reconfiguration after client/server mutual movements.

In more details, we measured these values in different testing conditions and we observed their dependence on several factors. They primarily relate with the interval T elapsed between the server loss of connectivity in the original locality and the complete re-establishment of server connection to the new location. The interval T, in its turn, mainly depends on the time employed by the peer that hosts the server component to cross the physical distance between the coverage areas of the two localities (movement time T_M) and on the time required by IEEE 802.11 devices to manage the communication handoff at the lower layers (handoff time T_H).

According to the measured values of T in several conditions, we decided to consider two main configurable parameters for COMMAND, which affect the performance of the election and server search protocols. The first, *IAPTime*, measures the election timeout value, i.e., the amount of time waited by a proxy candidate to announce its election. The second, *SearchPeriod*, in the case of server delays in reconnecting to the network, represents the time interval between successive proxy search attempts. COMMAND administrators can trade middleware promptness and generated traffic by a careful tailoring of *SearchPeriod* interval. In fact, by lowering that interval, the proxy is faster in promptly finding the server. However, if the server remains unreachable for a long time and no equivalent service component can be discovered, the overhead increases. In general, no static prediction can foresee a specific and optimized value. COMMAND provides a self-tuning mechanism that adapts parameters to server unavailability history. *SearchPeriod* and *IAPTime* have initial intermediate values specified by the administrator in an XML configuration file; they are automatically updated by monitoring the server behavior.

Table 1 reports values obtained by assuming that the server remains active during its movement. The proxy component is not able to find the server at the first search attempt because the movement (T_M) and the handoff (T_H) times required for the server reconnection to the new locality are longer than the interval to sense its movement and to elect the proxy in the old locality. Since the inactivity interval is low, it is worth to set also the *IAPTime* and *SearchPeriod* parameters to low values.

TABLE I. TEST CONDITIONS: IAPTIME = 200ms; SEARCHPERIOD = 200ms

Parameter	Average Value (ms)	Std. Dev. (ms)
Service Unavailability on Client	788.6	11.01
Search Time on Proxy	430.4	9.18

TABLE II. TEST CONDITIONS: SERVER INACTIVITY INTERVAL = 2s; IAPTIME = 1s; SEARCHPERIOD = 1s

Parameter	Average Value (ms)	Std. Dev. (ms)
Service Unavailability on Client	2270.4	23.10
Search Time on Proxy	1045.6	33.68

Table 2, instead, shows the performance measured when the server remains inactive for 2s after its disconnection from its original locality. Given that high inactivity interval, it is likely that the proxy is ready to find the server just as soon as it reconnects. This explains why the overhead values obtained by subtracting the inactivity interval from the *Service Unavailability on Client* (which represents a rough estimate of the time spent by COMMAND to rebind the client/server connections, once the server component has reconnected in the new locality) in the two scenarios are different. In this case, it is better not to pay a high overhead to promptly react to server movement. Thus, *IAPTime* and *SearchPeriod* can be set to relatively high values.

VI. RELATED WORK

MANET have recently attracted the interest of several research activities in both industry and academia [1]. First investigations have addressed the novel challenging communication issues, primarily to face the instability due to terminal mobility and infrastructure lack. On the one hand, some proposals provide solutions for the network-layer autonomous configuration of nodes that dynamically and unpredictably join MANET localities. The common goal of these solutions is to provide a temporary IP lease, without requiring explicit operations by network administrators [12]. On the other hand, relevant research activities have investigated novel MANET-specific solutions for multi-hop routing. A possible taxonomy of MANET routing protocols has been proposed in [13]: most common solutions are topology-based on-demand ones that determine routes to destination only by need when required by source nodes.

However, as already stated, MANET dynamicity does not affect only network-layer aspects but also significantly complicates application design and implementation. To support application development over mobile systems, some research activities are moving to extend traditional socket programming to embed mobility-related connection re-qualification [14]. In addition, other research is addressing the definition of novel tuple-based programming models suitable for highly dynamic MANET environments, primarily to support time/space decoupled coordination among distributed components. For instance, LIME provides coordination for MANET software

components by dynamically aggregating tuple spaces of co-located components in federated tuple spaces [15]. TOTA, instead, extends the LIME model with the idea of moving tuples in MANET deployment environments and of dynamically modifying tuple content according to some associated propagation rules [16].

To the best of our knowledge, there are not other application-level middlewares yet based on lightweight code mobility to support the automatic re-qualification of resource bindings. In that sense, our approach provides an original perspective of the field. However, the relevance of the addressed topic is recognized and some first proposals, which adopt design guidelines similar to COMMAND, are starting to emerge. For instance, to face MANET device heterogeneity, the CONNECTED project proposes application level proxies to carry on tasks that resource-constrained devices cannot perform [17]. In addition, the recent Expeerience support proposes the exploitation of mobile code techniques, in particular of mobile agent ones, to increase the flexibility of the MANET middleware [18].

VII. Conclusions and On-Going Work

Distributed applications over MANET require flexible and mobile middleware solutions capable of properly handling the frequent variations of locally reachable device/service components during service provisioning. In addition, the intrinsic complexity of the MANET scenario motivates a clear separation of concerns between the client/server application logic and the support solutions in charge of handling the discovery, automatic rebinding, and request routing to mobile service components. Novel MANET middlewares should exploit lightweight code distribution to effectively provide this separation and to achieve the level of flexibility and reusability suited to these highly dynamic network environments.

Our first experiences stemming from the deployment and testing of the COMMAND middleware have shown that a highly dynamic support infrastructure based on proxies and code mobility can significantly facilitate the design and implementation of MANET applications with feasible performance results, thus potentially leveraging the promising market of services for Personal Area Networks. These encouraging results are stimulating further research to extend the middleware in two main directions. First, we work on the full integration of our middleware prototype with other multi-hop routing protocols available in literature (and on the related performance evaluation). Secondly, we intend to explore solutions to dynamically establish proxy-to-proxy inter-locality chains. This perspective can speed up and facilitate the search of non-local servers, and can significantly improve the COMMAND scalability when deployed over large-scale MANET scenarios.

Acknowledgements

Work partially supported by the Italian MIUR within the FIRB WEB-MINDS Project and by the Italian CNR within the Strategic IS-MANET Project.

References

[1] J. Macker, S.Corson, "Mobile Ad-hoc Networks (manet)", 1997, http://www.ietf.org/htmlcharters/manet-charter.html.

[2] I. Chlamtac, M. Conti, J. J.-N. Liu, "Mobile ad hoc networking: imperatives and challenges", Elsevier Ad Hoc Networks, vol.1, pp.13-64, July 2003.

[3] J. Bolliger, T. Gross, "A Framework-based Approach to the Development of Network-aware Applications", IEEE Transactions on Software Engineering, vol. 24, pp. 376-390, May 1998.

[4] C. E. Perkins, E. M. Royer, "Ad Hoc On-demand Distance Vector Routing", 2nd IEEE Workshop on Mobile Computing Systems and Applications, WMCSA '99, pp.90-100, February 1999.

[5] A. Fuggetta, G. P. Picco, and G. Vigna, "Understanding Code Mobility", IEEE Transactions on Software Engineering, vol. 24, pp.342-361, May 1998.

[6] P.K. McKinley, U.I. Padmanabhan, N. Ancha, S.M. Sadjadi, "Composable Proxy Services to Support Collaboration on the Mobile Internet", IEEE Transactions on Computers, vol. 52, pp. 713-726, June 2003.

[7] S.S. Yau, F. Karim, W. Yu, W. Bin, S.K.S. Gupta, "Reconfigurable Context-Sensitive Middleware for Pervasive Computing", IEEE Pervasive Computing, vol. 1, pp. 33-40, July-September 2002.

[8] P. Bellavista, A. Corradi, R. Montanari, C. Stefanelli, "Context-aware Middleware for Resource Management in the Wireless Internet", IEEE Transactions on Software Engineering, vol. 30, pp. 1086-1099, December 2003.

[9] E. Wilde, "XML Technologies Dissected", IEEE Internet Computing, vol. 7, pp. 74-78, September-October 2003.

[10] Imperial College – Ponder, http://www-dse.doc.ic.ac.uk/Research/policies/ponder.shtml.

[11] P. Bellavista, A. Corradi, C. Stefanelli, "Java for On-line Distributed Monitoring of Heterogeneous Systems and Services", The Computer Journal, Oxford University Press, vol. 45, pp. 595-607, November 2002.

[12] S. Nesargi, R. Prakash, "MANETconf: configuration of hosts in a mobile ad hoc network", Joint Conference of the IEEE Computer and Communications Societies, INFOCOM '02, pp. 1059-1068, June 2002.

[13] E. M. Royer, C. Toh, "A review of current routing protocols for ad hoc mobile wireless networks", IEEE Personal Communications, vol. 6, pp. 46-55, April 1999.

[14] U. Saif, J. M. Paluska, "Service-oriented Network Sockets", USENIX Int. Conference on Mobile Systems, Applications and Services, MobiSys 2003, pp.159-172, May 2003.

[15] A. L. Murphy, G. P. Picco, G.-C. Roman, "LIME: a middleware for physical and logical mobility", IEEE Int. Conference on Distributed Computing Systems, ICDCS-21, pp. 524-533, April 2001.

[16] M. Mamei, F. Zambonelli, "Programming Pervasive and Mobile Computing Applications with the TOTA Middleware", IEEE Int. Conference on Pervasive Computing and Communications, PerCom 2004, pp.263-276, March 2004.

[17] Swedish Institute of Computer Science – CONNECTED, www.sics.se/cna/connected/.

[18] M. Bisignano, A. Calvagna, G. Di Modica, O. Tomarchio, "Expeerience: a JXTA middleware for mobile ad-hoc networks", IEEE Int. Conference on Peer-to-Peer Computing, P2P '03, pp. 214-215, September 2003.