MOBIDIS: A Pervasive Architecture for Emergency Management

Massimiliano de Leoni Fabio De Rosa Massimo Mecella

Università di Roma "La Sapienza"
Dipartimento di Informatica e Sistemistica - DIS (sede decentrata di Latina)
{deleoni,derosa,mecella}@dis.uniroma1.it

Abstract

Current mobile and pervasive technologies (e.g., PDAs, GPRS/UMTS and WiFI connections, etc.) enable the development of adaptive peer-to-peer software infrastructures for supporting collaborative work of human operators in emergency/disaster scenarios. In this paper, we present a novel architecture, named MOBIDIS (Mobile @ DIS), currently under development in the context of an IST research project, in which operators, equipped with hand-held devices, are coordinated by a workflow management system able to adaptively change the process schema in order to cope with anomalies. Some preliminary experimental results are also presented.

1. Introduction

The widespread availability of network-enabled hand-held devices (e.g., PDAs with WiFi - the 802.11x-based standard - capabilities) has made the development of pervasive computing environments an emerging reality. This in turn enables building an adaptive peer-to-peer software infrastructure for supporting collaborative work of human operators in emergency/disaster scenarios. Each team member is equipped with hand-held devices (PDAs) and communication technologies, and should carry on specific tasks. In such a way we can see the whole team as carrying on a process.

The team constitutes a MANET (Mobile Ad hoc NETwork, [1]) in which the team leader's device coordinates the other team members devices. In MANETs, mobile devices communicate one with another via wireless links without relying on an underlying infrastructure: each device acts as an endpoint and as a router forwarding messages to devices within radio range. MANETs are a sound alternative to infrastructure-based networks whenever an infrastructure is no longer available, or can't be used, as in emergency scenarios [7].

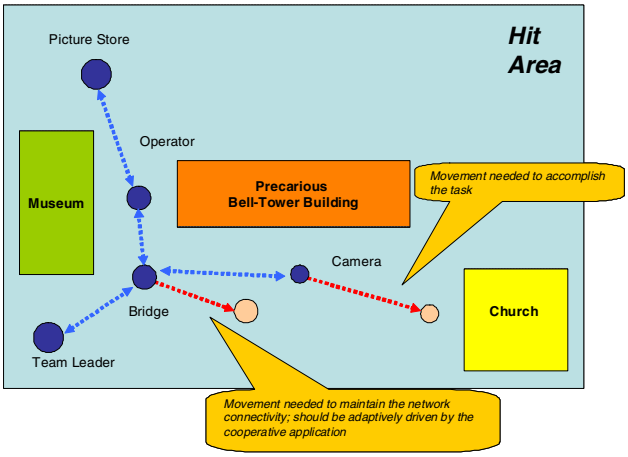


Figure 1. Critical situation and adaptive management

As an example, after a disastrous event (e.g., an earthquake) a team is sent to the disaster area. Team members, equipped with mobile devices need to document damage directly on a situation map so that following activities can be scheduled (e.g., reconstruction jobs). In this situation, matching new pictures with previous ones might be useful. So, the device with the high-resolution camera and the device with the older stored pictures must be connected. But in a scenario such as the one in Figure 1, the camera-equipped device's movement might result in its disconnection from the other devices. A pervasive architecture should be able to predict such situations in order to alert the coordination layer. The coordination layer, in turn, would direct a "bridge" device (member called `bridge` in the figure) to follow the device that's going out of range,

maintaining the connectivity and ensuring a path between the devices.

The purpose of this paper is to focus on the coordination layer of our pervasive architecture (initially proposed in [5, 7]). Coordination layer's purpose is to coordinate actors belonging to the same team in order to carry on processes by assigning to specific actors specific tasks to be carry out through specific applications running on hand-held devices. All such applications typically require continuous inter-device connections (e.g., for data/information sharing, activity scheduling and coordination, etc.) and, as argued, this matter is not generally guaranteed in MANETs, due to the high mobility of the nodes in the network. So the system has to be adaptive, able to adapt correctly processes to changing contexts where they are carried on.

2. Related Work

In recent years, research in the MANET area has mainly focused on the development of appropriate routing protocols, security and reliability of the communications, methods for energy preservation, and other issues on the lower four ISO/OSI layers [3–5]. Effective routing in ad hoc networks is still an actively-addressed open problem, with some interesting proposals presented in the literature (e.g., DSR - Dynamic Source Routing, AODV - Ad hoc On demand Distance Vector, Z–RP - Zone Routing Protocol, etc.). Researchers in this area assert that a sound technical basis for MANETs exists and it is thus time to start thinking about how to support applications based on MANETs.

The computerized facilitation or automation of a process, in whole or part, is named workflow. A Workflow Management System (WfMS) is a system that completely defines, manages and executes workflows [2]. A workflow process definition (workflow schema) is made up of a set of tasks, atomic piece of work, to be performed according to a specified scheduling (i.e., routing or control flow), by actors, such as humans and software applications/services. Each task needs an actor covering a given set of *roles* which define organizational requirements and needed skills [8]. A process is said to be *well-structured* if it is only involves the six well-behaved control structures, i.e., sequence, selection, parallel, loop, begin and end structures and each split is complemented by a join [2].

The critical issue of current WfMSs is the lack of support for adaptiveness; most of WFMSs (both commercial products and academic prototypes) handling adaptiveness assume the presence of an expert who decides which changes have to be applied to workflow schemas. But in MANET environments the assumption of an expert whose only purpose is disconnection handling is not acceptable. So, we must investigate how automatically decide, first, the inadequacy of a process instance, and, then, which changes on the corresponding workflow definition must be applied to handle disconnections.

In general the issue of adaptive workflow management [9] is still open. Relevant work are the e-Flow system [10], in which the issue of manually modifying workflow schemas is addressed, together with automatically migrating active process instances to the new schema. AGENTWORK [11] is one of the few examples of workflow system where adaption is not manual, but automatic, on the basis of a rule-base approach.

But in MANETs, the adaption should be carried out in a very frequently changing environment due to the MANET peculiarities. On the contrary, previous approaches are targeted to Web-based workflows (particularly workflow schemas composed by different Web services), in which modifications of the schemas are less frequent, but the number of running instances is very high.

A lot of relevant research work and projects about the issue of supporting collaborative work in emergency situations, by using pervasive and mobile technology, can be found in literature. For example, SHARE[1] develops advanced mobile services to support rescue forces during their operation. The goal of EGERIS[2] is to provide Civil Protection organizations (and different actors in Emergency Management) with information and communication technologies that improve their overall efficiency during the preparedness and the response phases of a crisis. In all presented works, the peer-to-peer paradigm is not used, but we believe that such a paradigm together with service-oriented architectures (SOA) and MANETs are fundamental for supporting collaborative work in emergency and disaster scenarios.

3. A Pervasive Architecture for MANET

In this section, we report our approach to the workflow management on MANETs. We assume each device includes hardware that lets it know its "communication" distance from devices within radio range. This isn't a very strong assumption, because specific techniques and methods are easily available (e.g., TDOA - time difference of arrival, SNR - signal-to-noise ratio, GPS hardware, etc.). At start-up, all devices are connected (that is, each device has a path, possibly multihop, to any other de-

[1]SHARE FP6-004218 project: `http://www.ist-share.org`.
[2]EGERIS IST-2000-28345 project: `http://www.egeris.org`.

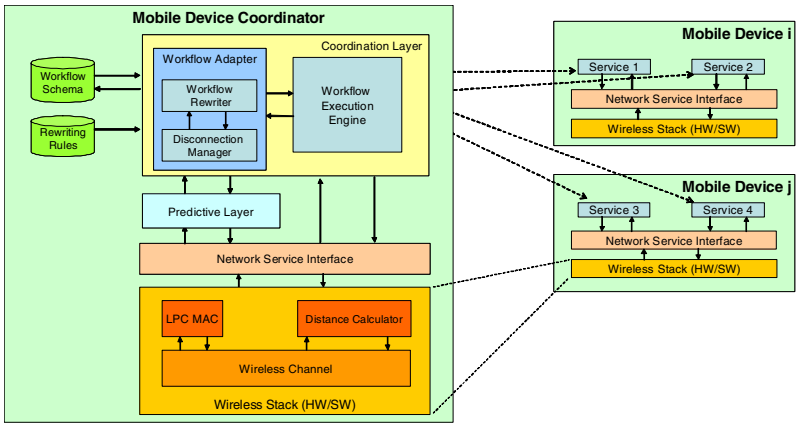


Figure 2. The proposed architecture for supporting cooperative work on MANETs

vice). Each device doesn't have to be within range of all other devices; that is, a tight (one-hop) connection is not required. Many routing protocols, as those ones described in Section 2, guarantee that each device requires only a loose connection.

The proposed approach combines local connection management among devices with global management of both network topology and task assignment. Local connection management consists of monitoring and checking one-hop communications between a device and its neighbors. It's realized as special services running on hand-held devices that implement techniques for estimating and calculating distances and relative positions between a specific device and its direct neighbors. Global management maintains a consistent state of the network and of each peer in the network. It manages the network topology (and its predicted next states) and the tasks each peer is in charge of, as well as services that peers offer (that is, it provides a service registry). On the basis of that information, the coordinator applies algorithms for choosing a bridge and/or executes workflow task reassignment when needed. Figure 2 shows the proposed architecture: each device has a *wireless stack* consisting of a wireless network interface (the *wireless channel* and *LPC MAC* modules) and the hardware for calculating distances from neighbors (*distance calculator* module). On top, a *Network Service Interface* [5,6] offers to upper layers the basic services for sending and receiving messages (through multihop paths) to and from other devices, by abstracting over the specific routing protocols.

Offered services (i.e., specific applications supporting tasks of the devices' human users) are accessible to other devices and can be coordinated and composed cooperatively. Some of these services are applications that don't require human intervention (for example, an image-processing utility). Others act as proxies for humans (for example, the service for instructing human users to follow a peer is a simple GUI that alerts the user by displaying a pop-up window on his/her device and emitting a signal).

In contrast, the coordinator device presents the *Predictive Layer* on top of the Network Service Interface, signalling any probable disconnection to the upper *Coordination Layer*. The Predictive Layer implements a probabilistic technique which can predict if all devices will still be connected in the next instant. For details about the predictive algorithm (together with its experimental results), please see [7].

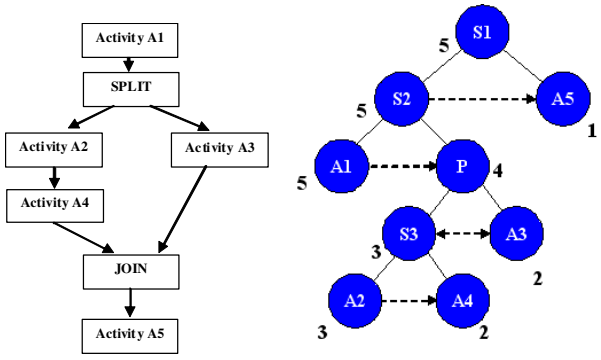
The coordination layer manages situations when a peer is going to disconnect, by applying algorithms for choosing a bridge, and by executing workflow schema restructuring and workflow task reassignment when needed (e.g., it assigns the activity "follow peer X" to the selected bridge).

4. Adaptive Coordination Layer

4.1. Collaborative Process Model

The first critical issue to build a WfMS is choosing a model to describe processes and theirs possible structure. For our goals, we have decided to use directed graph as process model and to handle only well-structured processes [2]. The most important nodes of process graph are tasks, atomic piece of work. Each task, identified by name, can have related a recovery task, when it is possible to achieve its logical undo or at least to minimize effect of its execution. A service is

associated to each task. The actors willing to perform the task have to provide that service. Moreover, each task node holds a reference to next node in the graph to be reached when its execution completes. An example of graph process model is depicted in Figure 3(a).



(a) Graph model (b) Process tree

Figure 3. Example of Graph Process Model and corresponding Process Tree

The adaptive WfMS we have designed needs some information at run-time: *(i)* the knowledge of the set of actors joined in every instant to the team; *(ii)* the set of services offered by each actor, so to establish which task each actor can perform; *(iii)* the state of every task of each process instance. The "bridging" actions needed to keep the MANET connected can be seen as *supporting* tasks associated to primary ones (i.e., tasks defined originally in the process schema), added or deleted during the execution of the process instance.

4.2. Bridge and Priority Algorithms

When a peer is going to disconnect, the coordination layer has to apply a specific algorithm for choosing a possible bridge. The algorithm prefers as possible bridges the neighbors that are not performing tasks. Indeed, if the selected neighbor carries out a task, such a task is rolled-back and productivity decreases. If each neighbor is carrying out a task, then it is chosen the one performing the task with the lowest priority[3]. If two or more actors perform tasks with the same lowest priority, the one is preferred with the smallest number of neighbors. The bridge role likely leads to movement of the node and this might cause new disconnections. By selecting a node with the lowest number of neighbors, the probability of new disconnections is minimized. In the end, the nearest neighbor is preferred.

An important issue of the bridge algorithm is computing task priorities. In our model, priority is computed at design-time and is then updated at run-time (after process restructuring). The priority has to reflect the process structure (causal dependency): the purpose is to assign higher priority to those tasks whose completion lets a greater number of tasks to be enabled for possible execution.

The algorithm for computing priorities is based on a n-ary tree, named *process tree*, that can be built iteratively. A well-structured process is decomposable in parts. Each one in turn can be considered as a well-structured process; therefore, each part can be further decomposed in smaller parts and so on up to elementary ones, the tasks. Each tree node is a process (elementary or not) whose children are nodes representing the parts it can be decomposed in. Let's consider the process tree depicted in Figure 3(b) which corresponds to process in Figure 3(a). If a single arrow exists between two sibling A and B nodes, it means that the process is decomposed into a sequence of two parts A and B. If a double arrow exists between two siblings A and B, it means that the process is decomposed such that A and B are performed in parallel or any arbitrary order. In the end, if there is no arrow between two siblings A and B, that means process is decomposed such that there is a selection in carrying out between A and B, according to some conditions (the process instance in Figure 3(a) does not contain such a situation).

Starting from the process tree, it's possible to assign to each node a weight. Initially, a weight equal to 1 is assigned to every leaf node. For each non-leaf node P a weight is assigned according to how P is decomposed in. If P is a sequence-type node, then its weight is obtained as the sum of its children nodes' weights; if P is a parallel-type node, then its weight is obtained as the sum of its children nodes' weights plus 1; finally, if P is a selective-type node, then its weight is the maximum value, taken from weights of its children nodes, plus 1. After, a weight adjustment is required: for each sequence-type node N_i, if its left child node $N_{i_{left}}$ has a weight less than N_i, then $N_{i_{left}}$'s weight is increased in the quantity $N_i.weight - N_{i_{left}}.weight$, as well as all nodes belonging to the sub-tree whose root is $N_{i_{left}}$. The algorithm computes finally node weights, that is priorities of the process parts which nodes correspond to. Specifically, leaves' weights are the priorities of elementary processes, i.e. tasks. In Figure 3(b), node labels are weights which algorithm computes.

[3]Indeed, rolling-back task with the lowest priority should bound inefficiency.

4.3. Process Schema Restructuring

We can now describe the restructuring rules for process schemas. Mainly, these rules are applied when a task needs a supporting counterpart.

Let's consider a generic t task and suppose t's performing actor a is going to disconnect. In this case, "follow a" supporting task is needed for t, so it can progress. Please note that, anyway, t task is performed by a and it is not deferred to a bridge node; the bridge is used only to keep connectivity of a with the MANET.

Let f be this supporting task. The f introduction requires process schema restructuring. Restructured schema is obtained by adding f in parallel with task t. Since f's completion enables as many tasks as t's one, f's priority is set equal to t's one.

The introduction of a supporting task causes priorities to change only for tasks coming before the introduction point.

5. Experimental Results

In order to carry out experiments and to test our pervasive architecture, we developed an emulation system based on NS2. More details are given in [12].

Experiments in this context are influenced by the initial positions of nodes and objects, and by the graphs of the processes. In our preliminary experiments, nodes, obstacles and other objects were manually put in the map at design-time by using the GUI of the emulation system, and process schemas were chosen with both loops and AND/OR splits.

The purpose of the first part of experiments is to tune some parameters of the algorithms. Once parameters has been tuned, we have performed more deep experiments. Details are presented in [12].

The first tuned parameter is the *polling time*, i.e., the shortest time between two corrective actions; an higher value means more reactivity in doing corrective actions. The second parameter is β, i.e., the fraction of the radio-range the predictive technique doesn't signal a disconnection anomaly. As an example, in IEEE 802.11 with 100 meters of radio-range, β equal to 0.3 means that for a communication distance of 70 meters the prediction algorithm signals a probable disconnection.

β	**0.3**	**0.5**	**0.7**
polling time 3 sec	1%	0.09%	0.02%
polling time 5 sec	32%	4%	0,88%

Table 1. Experimental results.

The choices for parameter tuning are depicted in Table 1, varying polling time between 3 and 5 seconds and β between 0.3 and 0.7.

Preliminary, let's note that processes, in our experiments, are carried out for each value of each parameter; this is as a MANET node, once disconnected, is supposed to move in a random way inside the map, and eventually it comes back in the MANET. Therefore, the interesting result to be evaluated is how much processes stall during their execution. A process is considered stalled if it cannot progress; this happens if next tasks need to be performed only by nodes that are currently disconnected from the MANET. The process will be able to progress when and only when at least one of those nodes comes back in the MANET.

Values shown in Table 1 are the ratio between the stalling time and the total time (stalling time + effective execution time). Smaller values of β or greater polling time means less, respectively in number and in frequency, corrective actions, that is more movement freedom. A few freedom brings to a greater number of rollback actions and, so, to inefficiency in process progress. The most free experimental situation (β and polling time equal, respectively, to 0.3 and 5 sec.) shows stalling time to be unacceptably around a third of total time. This result does not depend on the scenario, as a mean value has been brought out on various settings for nodes and schemas.

The result suggests that less frequent the corrective actions are, the higher the stalling time is. So the extreme circumstance is when corrective actions are so much infrequent that they are absent. In that case, corresponding to no disconnection handling, the process stalling time should be inclined to total time and, so, to process starvation.

6. Conclusion and Future Work

In this paper, we have presented a novel pervasive architecture suitable in emergency scenarios for workflow management on MANETs, that through *(i)* a basic predictive layer for disconnection anomalies, and *(ii)* an adaptive coordination layer, is able to change the process schemas when disconnection anomalies are raised. We discussed the basic techniques developed and some preliminary experimental results.

In future work, we are going to further refine such basic techniques. Moreover, we are going to address the issue of the approach's fault tolerance: currently our approach doesn't cope with sudden downs of devices, which might be frequent in emergency scenarios and are critical if they affect the coordinator node.

We also plan to evolve the coordination layer from

a centralized to a distributed one (i.e., having a subset of devices act as coordinators). At the moment, the centralized architecture might be a bottleneck, but the current dimensions of a typical MANET for the considered scenarios (tens of devices) don't pose critical scalability issues.

Finally, our results are based on synthetic data, and thus are only a preliminary validation of our approach. Future work will be devoted to validate our approach in real scenarios.

Acknowledgements. This work is partly supported by the Italian MIUR, through the FIRB 2001 project *MAIS* - `http://www.mais-project.it`, and by the European Commission through the FP6-2005-IST-5-034749 project *WORKPAD*.

The authors would like to thank Fabio D'Aprano, an undergraduate student of the Faculty of Computer Engineering at Latina, University of Rome "La Sapienza", for his considerable contribution in building the experimental architecture.

References

[1] D.P. Agrawal, Q.A. Zeng: *Introduction to Wireless and Mobile Systems*, Thomson Brooks/Cole, 2003.

[2] W.M.P. van der Aalst, K. van Hee: *Workflow Management: Models, Methods, and Systems*, MIT Press, 2001.

[3] J. Kong, X. Hong, Y. Yi, J.S. Park, J. Liu, M. Gerla: A Secure Ad-hoc Routing Approach using Localized Self-healing Communities. Proc. *ACM MobiHoc'05*.

[4] N.H. Vaidya: Mobile Ad Hoc Networks: Routing, MAC and Transport Issues. Tutorial, `http://www.crhc.uiuc.edu/~nhv`, University of Illinois at Urbana-Champaign, USA, July 2004.

[5] F. De Rosa, V. Di Martino, L. Paglione, and M. Mecella: Mobile Adaptive Information Systems on MANETs: What We Need as Basic Layer?. Proc. *WISE 2003 Workshop on Multichannel and Mobile Information Systems (MMIS'03)*.

[6] F. De Rosa, M. Mecella: Designing and Implementing a MANET Network Service Interface with Compact .NET on Pocket PC. Proc. *3rd International Conference on .NET Technologies (.NET 2005)*.

[7] F. De Rosa, A. Malizia, M. Mecella: Disconnection Prediction in Mobile Ad hoc Networks for Supporting Cooperative Work. *IEEE Pervasive Computing* 4(3), 2005.

[8] W.M.P. van der Aalst, A.H.M. ter Hofstede, A.P. Barros: Workflow Patterns. *Distributed and Parallel Database* 14, pp. 5–51, 2003.

[9] M. Voorhoeve, W.M.P. van der Aalst: Ad-hoc Workflow: Problems and Solutions. Proc. *8th DEXA Conference*, 1997.

[10] F. Casati, M.C. Shan: Dynamic and Adaptive Composition of *e*-Services. *Information Systems* 6(3), 2001.

[11] R. Müller, U. Greiner, E. Rahm: AGENTWORK: A Workflow-System Supporting Rule-Based Workflow Adaptation. *Data and Knowledge Engineering* 51(2), pp. 223–256, 2004.

[12] F. D'Aprano, M. de Leoni, F. De Rosa, M. Mecella: On-line Appendix to the Paper "MOBIDIS: A Pervasive Architecture for Emergency Management". Univ. Roma "La Sapienza", Dipartimento di Informatica e Sistemistica, Tech. Rep. 04/2006, `http://www.dis.uniroma1.it/~deleoni/documents/AppendixDMC2006.pdf`, 2006.