Computing and Informatics, Vol. 29, 2010, 537–555

ENABLING INFORMATION GATHERING PATTERNS FOR EMERGENCY RESPONSE WITH THE OPENKNOWLEDGE SYSTEM

Gaia Trecarichi, Veronica Rizzi, Maurizio Marchese

Department of Information Engineering and Computer Science
University of Trento
Via Sommarive 14, I-38123 POVO (TN), Italy
e-mail: {gtrecari, vrizzi, marchese}@disi.unitn.it

Lorenzino Vaccari

Spatial Data Infrastructure Unit
European Commission – Joint Research Center
via Enrico Fermi 2749, 21027 Ispra (VA), Italy
e-mail: lorenzino.vaccari@jrc.ec.europa.eu

Paolo Besana

Laboratoire d'informatique médicale
Université de Rennes I
2 Avenue du Pr. Leon Bernard, 35043 Rennes Cedex, France
e-mail: paolo.besana@univ-rennes1.fr

Revised manuscript received 24 March 2010

Abstract. Today's information systems must operate effectively within open and dynamic environments. This challenge becomes a necessity for crisis management systems. In emergency contexts, in fact, a large number of actors need to collaborate and coordinate in the disaster scenes by exchanging and reporting information with each other and with the people in the control room. In such open settings, coordination technologies play a crucial role in supporting mobile agents located in areas prone to sudden changes with adaptive and flexible interaction patterns.

Research efforts in different areas are converging to devise suitable mechanisms for process coordination: specifically, current results on service-oriented computing and multi-agent systems are being integrated to enable dynamic interaction among autonomous components in large, open systems. This work focuses on the exploitation and evaluation of the OpenKnowledge framework to support different information-gathering patterns in emergency contexts. The OpenKnowledge (OK) system has been adopted to model and simulate possible emergency plans. The Lightweight Coordination Calculus (LCC) is used to specify interaction models, which are published, discovered and executed by the OK distributed infrastructure in order to simulate peer interactions. A simulation environment fully integrated with the OK system has been developed to: (1) evaluate whether such infrastructure is able to support different models of information-sharing, e.g., centralized and decentralized patterns of interaction; (2) investigate under which conditions the OK paradigm, exploited in its decentralized nature, can improve the performance of more conventional centralized approaches. Preliminary results show the capability of the OK system in supporting the two afore-mentioned patterns and, under ideal assumptions, a comparable performance in both cases.

Keywords: Interaction modeling, P2P coordination, crisis management, agent-based simulation

Mathematics Subject Classification 2000: 68T35, 68T27, 68N17, 68M14

1 INTRODUCTION

In crisis management, it is important to coordinate emergency response[1] activities in an effective way. At present, most of the information management infrastructures required for dealing with emergencies are based on centralized architectures that (i) are specifically designed prior to the emergency, (ii) gather centrally the available information, (iii) distribute it upon request to the appropriate agents (e.g., emergency personnel, doctors, citizens). While such infrastructures provide a number of significant advantages (e.g., quality control, reliability, trustworthiness, sustainability, etc.), they also present some well-known intrinsic problems (e.g., physical and conceptual bottlenecks, communication channel overloads, single point of failure). Alternative solutions are currently being explored, studied and analyzed to support data sharing also in the absence of a centralized infrastructure [6, 5]. In this study, we explore the flexibility and adaptability of the framework developed within the OpenKnowledge project[2]. This framework provides a distributed infrastructure, that enables peers to find and coordinate with each other by publishing, discovering and executing multi party conversational protocols.

[1] In the rest of this paper, we will refer to "emergency response" as "e-Response".

[2] `http://www.openk.org`.

In this paper, the proposed OpenKnowledge infrastructure is used to explore its capability to support both centralized and decentralized information-gathering patterns in open environments. For this purpose, we built a simulation-based testbed fully integrated with the OK platform. The final goal of such virtual environment is to evaluate this framework in the e-Response domain. In particular, we implemented an e-Response simulation environment through which existing emergency plans based on real-data are modelled and simulated. Moreover, a suite of experiments has been designed and run to evaluate the performance of the OK e-Response system under specific assumptions. Preliminary results show the system's capability of supporting the two afore-mentioned architectures and a comparable performance in both cases. The idea to test with simulations the effectiveness of different data management architectures within e-Response settings is not new, as shown by the agent-based simulations developed in [7, 8, 9].

The rest of the paper is organized as follows: in Section 2, the fundamentals of the OK system are presented. Section 3 presents the e-Response case study, based on realistic data and emergency plans, which is used to evaluate the OK system. In Section 4, the p2p simulation environment architecture is analysed and, in Section 5, the experimental testbed designed for the evaluation is presented; here the preliminary results of centralized vs. decentralized information management architectures are discussed. Conclusion and future work are given in Section 6.

2 THE OPENKNOWLEDGE SYSTEM

The OpenKnowledge (OK) system provides the underlying peer-to-peer infrastructure needed to run our experiments. The core concepts in OK are: (1) the interactions between agents, defined by interaction models; and (2) a distributed infrastructure, denoted as OK kernel, that supports the publishing, discovery, execution, monitoring and management of interaction models.

Interaction models[3] are written in the Lightweight Coordination Calculus (LCC) [2], an executable choreography language based on process calculus and specifically designed to express P2P style interactions within multi-agent systems. Interactions in LCC are expressed as the message passing behaviours associated with roles. The most basic behaviours are to send or receive messages, where sending a message may be conditional on satisfying a constraint (precondition) and receiving a message may imply constraints (postcondition) on the agent accepting it. LCC makes no commitment to the method used to solve constraints, so different participants might operate different constraint solvers (including human intervention). LCC clauses of some interactions relevant to our experiments are shown later in this paper (Figure 4, Section 4.1).

The OK kernel [1] provides the layer that assorted services and applications can use to interact using a choreography-based architecture able to deal both with the semantic heterogeneity of the actors and with their discovery. The IMs are

[3] Henceforth, we use the shorthand term "IM" in place of "interaction model".

published by the authors on a distributed discovery service (DDS) with a keyword-based description. A peer willing to perform a task (e.g., verifying the flood state in some area), searches for published IMs by sending a keyword-based query to the DDS which in turn collects them by matching the query and sends back the list to the peer; it then selects an interaction by comparing the constraints with its own capabilities; finally, it subscribes to one of the roles on the DDS. When, in a given interaction, all the roles are subscribed by at least one peer, a compatible team is formed and the interaction is executed.

The peer's capabilities are provided by plug-in components, called OpenKnowledge Components (OKCs) and consist of a set of Java methods. Figure 1 shows a snapshot of a network, when the roles in interaction IM1 are all subscribed by at least one peer. The peers have installed their OKCs locally: some of them can be found online (e.g., OKC1, OKC2 and OKC3), others might be private to a peer (e.g., OKC4).

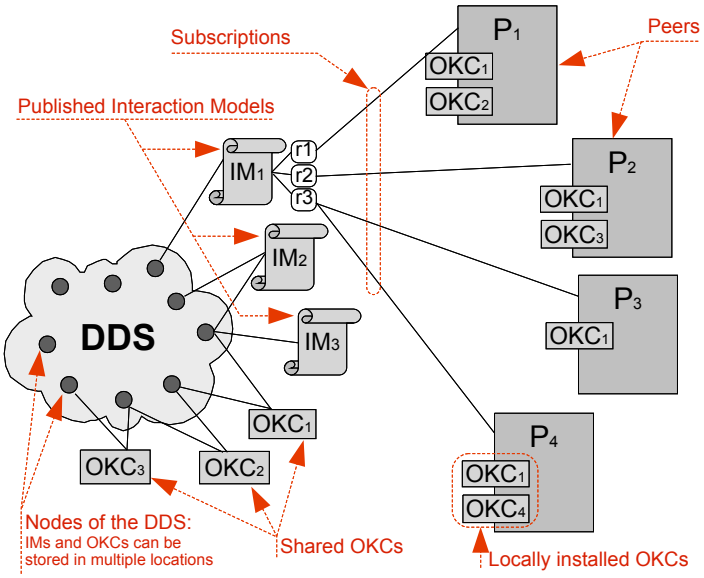


Fig. 1. OpenKnowledge architecture

3 EMERGENCY RESPONSE: A CASE STUDY

This section presents the case study used to evaluate the OK system: an e-Response scenario. In particular, we considered a flooding disaster in Trento (Italy). The work started from a preliminary analysis on this kind of disaster, resulted from documents related to the current flood emergency plan foreseen by the Trentino region and from interviews with experts. We selected emergency peers (e.g., firemen,

police, bus/ambulance agents, etc.), the main organization involved (e.g., Emergency Coordination Center, Fire Agency, Civil Protection[4], etc.), a hierarchy between the actors (e.g., emergency chief, subordinate peers, etc.), service peers (e.g., water level sensors, route, weather forecast and GIS services) and a number of possible scenarios, i.e., possible interactions among the agents and their assigned tasks. The peers can be distinguished into two main categories: *service peers* and *emergency peers*. While the former are basically peers providing services under request, the latter are peers often acting on behalf of human agents that are in charge of realizing the emergency plan.

Our analysis resulted in the modeling of the *pre-alarm* and the *evacuation* phases of the emergency plan. In the pre-alarm phase, an Emergency Monitoring System (EMS) continuously gathers data from water level sensors placed in strategic points (break points) along the river. It also checks weather information in order to enrich the data needed to predict the evolution of a potential flood. When a critical situation is registered, the automatic system notifies the emergency chief, who can then decide whether to enact the evacuation plan or not. The evacuation phase regards all the activities needed to move people to safe places. Here, the key peers are emergency peers, i.e., all the peers in charge of helping in the evacuation of citizens: emergency coordinators, firemen, government agencies (e.g., civilian protection department), real-time reporters (e.g., people, sensors). Of course, the emergency peers are supported by service peers such as route services and sensors scattered across the emergency area. The evacuation plan considered consists of agents (e.g., emergency subordinates such fire-fighters) moving to specific locations assigned by the chief. In order to move, they need to perform some activities: choosing a path to follow by asking a route service; checking if the path is practicable by interacting with the CP or with available reporters distributed in the area; proceeding along the path. The CP is able to serve requests on the blockage state of a given path, since it continuously gathers information from reporters (e.g., sensors) scattered around the emergency area and reporting the water level registered at their locations.

Figure 2 sketches the two phases involved in our case study. It shows the involved actors (denoted by round circles), their interactions and the kind of information exchanged. The smooth rectangle denotes the simulator, i.e., the virtual environment where all the peers act; obviously, it does not correspond to any entity in the reality, therefore, we do not describe it in this context. However, the simulator is essential for the simulation-based testbed and will be illustrated in detail in Section 4.2.

It also shows two different evacuation sub-scenarios: in both of them, an emergency subordinate (ES) needs to get information on route's practicability but while in one case (area above the red line) such peer gets route information by asking the CP, in the other one (area below the red line) it interacts directly with reporters (r1, r2, r5) physically present at the locations of interest. These two information-gathering patterns are referred to as *centralized* and *decentralized* strategies.

[4] Henceforth, we use the shorthand term "CP" in place of "Civil Protection".

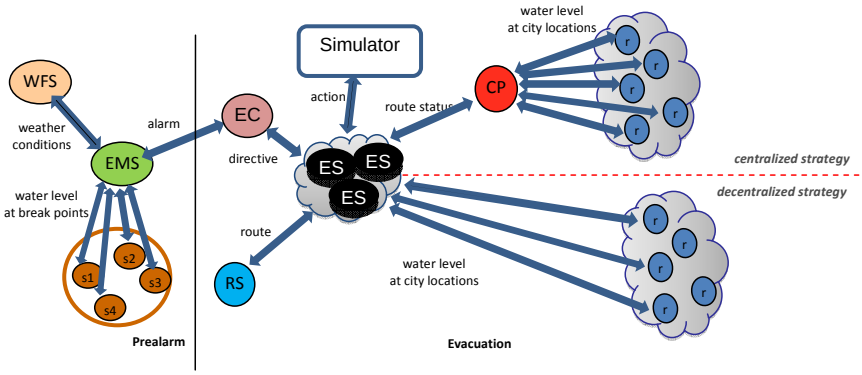


Fig. 2. e-Response case study: pre-alarm and evacuation phases

4 THE PEER-TO-PEER SIMULATION ENVIRONMENT

To fully use and test the OK system in the previously described e-Response scenario, we built a p2p simulation environment. The current e-Response simulation environment is based on the system presented in [4] and extends it both in a complete integration with the OK kernel and in the inclusion of a realistic flood-simulator. The testbed is used to evaluate IMs, coordination tasks and the diverse emergency information-gathering models; through simulations, it is possible to estimate how the platform could perform in a "real-world" emergency. In particular, the developed e-Response system is used to:

1. model and execute interactions among peers (e.g., institutional agents, individuals, sensors, web services) involved in emergency response activities;
2. provide feedbacks about the environment at appropriate moments, in a way that mirrors the real world (for example, an agent attempting to take a road will be informed at that moment if that road is blocked, and it can then share this information with other peers through the network);
3. visualise and analyze a simulated coordination task through a Graphical User Interface (GUI).

The e-Response system is composed of two main components: the peer network and the e-Response simulator. Figure 3 sketches its overall architecture. All peers are equipped with their own OKCs; black arrows represent LCC interactions either among simulator peers or between simulator and network peers; the grey arrows indicate interactions among network peers only.

4.1 The Peer Network

The peer network represents the group of agents involved in a simulated coordination task. An agent in the peer network can interact with other agents, perform some

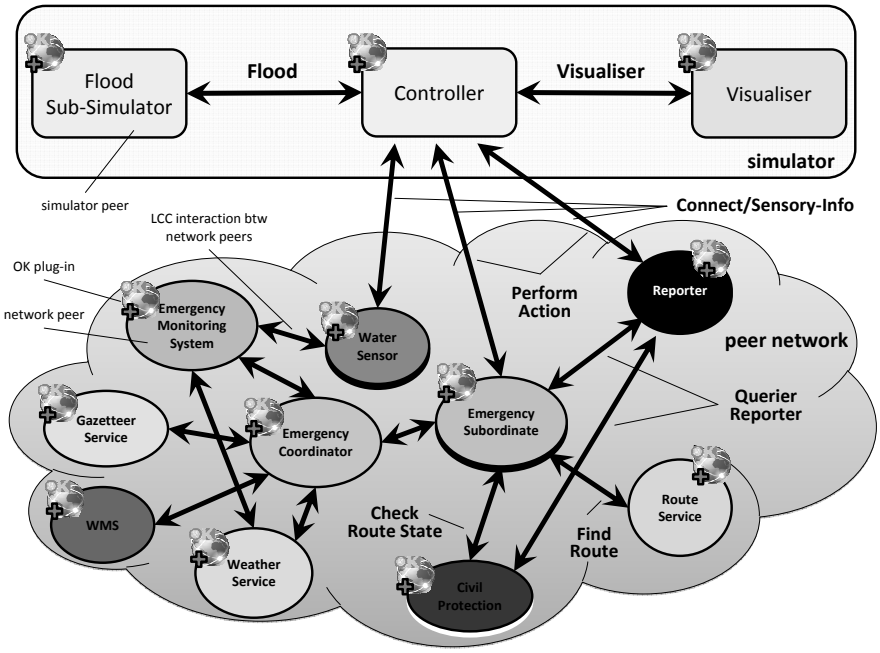


Fig. 3. The e-Response system's architecture

actions (e.g., move along a road) and gather information (e.g., sense the water level in its vicinity).

In order to perform an action or receive sensory information near its location, a peer must connect to the simulator by enacting the "Connect" IM. Once added to the simulation, the connected peer periodically receives sensory information from the simulator via the "Sensory-Info" IM; finally, to perform an action, a connected peer enacts the "Perform-Action" IM which models the action coordination with the simulator. The connected network peers are called *physical peers* (shaded ellipses in Figure 3).

Not all peers must connect to the simulator: *non-physical peers*, such as a route service that provides existing routes, do not need to communicate with the controller but only with other peers in the peer network. In the real world such peers would not actually be in the disaster area and could not affect it directly, but could provide services to peers that are there. Non-physical peers are represented as not shaded ellipses in Figure 3.

The interactions – modeled as LCC specifications – among the network peers which take place during the evacuation phase (see Figure 2) can be recaped as follows:

1. *Start evacuation*: the emergency coordinator *EC* alerts its subordinates to go to a specific location. The team-members prepare to satisfy the directive;

2. *Find a route*: an emergency subordinate *ES* requests a path connecting its current location to the destination from a route service *RS*;
3. *Check path conditions*: a subordinate *ES* relies either on the *CP* (centralized strategy) or on reporters dislocated in the interested area (decentralized strategy) to get information on the route's practicability;
4. *Gather info from reporters*: an agent (e.g., CP, ES) requests real-time water level information from a group of reporters.

The "Start evacuation" IM is the main one in the selected use case. It simulates the evacuation phase and can be used in all those situations where an emergency chief sends the directive of reaching specific locations to its subordinates. In short, an emergency subordinate ES receives an alert message from the chief and resolves some constraints in order to set the goal to be achieved (reach the goal destination G) and get the current position. The activities of ES thus evolve through three key roles: the *goal achiever* role which abstractly models the activity of searching for a path and moving towards the goal; the *free path finder* role which defines the operations needed to find a free path; the *goal mover* role which models the actions needed to move towards the goal destination. Figure 4 shows an LCC code snippet for two of the key ES roles.

The constraints specified in bold indicate that they are solved by enacting separate IMs. This is a key functionality of the OK platform, since it allows to write simple, modular and reusable LCC specifications. In particular, the constraint *request_path_state* enacts the "Check-Route-State" ("Querier-Reporter") IM when the centralized (decentralized) information-gathering strategy is adopted. The execution of both IMs is needed to get route condition information and then decide whether to go ahead or to find another path. However, while in the centralized scenario the CP acts as the only provider of such information, in the decentralized setting there is a direct interaction between an emergency subordinate *ES* and a set of reporters. In this last case, after having found a route, the peer subscribes to the "Querier-Reporter" IM, selects an appropriate group of reporters and start interacting with them. The selection of a suitable group of reporters is done by looking at the reporter's subscriptions returned by the DDS and that indicate their current locations.

4.2 The e-Response Simulator

The simulator is designed to represent the environment where all the involved agents act. It is composed of three modules which are themselves peers: the controller, the flood sub-simulator, and the visualiser (Figure 3). The controller regulates the simulation cycles and the management of the simulated agent activities; the flood sub-simulator reproduces the actual evolution of the 1966 flood in Trento; the visualiser stores simulation information used by the GUI to view a simulation run in a step-by-step way. The simulator does not interfere or help the coordination among network peers: it just simulates the real world.

```
a(emergency_subordinate,FF)::

alert(G) <= a(emergency_chief,FFC) then
     null <- set_goal(G) and get_current_position(CurrPos) then
        a(goal_achiever(CurrPos,G),FF)


a(goal_achiever(From,To),GA)::

(
  //moving peer already at destination
  null <- equal(To,From) and setGoalAchieved(To)

   or

   ( //try to find a free path
     a(free_path_finder(From,To,FreePath), GA) then

     //no free paths between From and To
     null <- FreePath=[] and setGoalUnreachable(To)
```

a)

```
a(free_path_finder(From,To,FreePath), FRF) ::

null <- find_path(From,To,Path) then
(
  //no paths are found
  null <- Path=[] and makeEmptyList(FreePath)

  or

 (
   //check if the path is free
   null <- request_path_state(Path,PathState) and
           path_free(PathState) then
      null <- assign(Path,FreePath)
 )

 or

  //search for an alternative path which is free
  a(free_path_finder(From,To,FreePath), FRF)
)
```

b)

Fig. 4. LCC fragments; a) "goal-achiever" role, b) "free-path-finder" role

Controller drives the simulation cycles and keeps track of the current state of the world. In order to achieve that, it needs to know what changes are happening to the world and updates its state accordingly. After updating its state, it also informs the relevant peers of these changes. The simulation thus evolves through cycles (or time-steps). A simulation cycle foresees two main operations:

- *Gathering changes*: the controller receives information about the changes that happened to the world:

1. it receives the disaster (e.g., flood) changes from the disaster sub-simulator via the "Flood" IM and
2. it serves requests of performing (move) actions with the "Perform-Action" IM (see Figure 3). In this latter interaction, the controller verifies whether certain actions are legal or not before they are performed, and if a certain action is illegal, the peer is informed of the reason of failure;

- *Informing peers*: the controller sends information about the changes that happened in the world:

1. it sends, at each time-step, local changes to each connected peer via the "Sensory-Info" IM and
2. it sends, to the visualiser, information on –
   (a) the locations of all connected peers;
   (b) the status of the reporter peers (e.g., available, responding to requests) and
   (c) the water level registered; here, the "Visualiser"IM is used.

Before a simulation cycle initiates, some preliminary activities are performed: establish key parameters (e.g., number of simulation cycles, timeouts, water level thresholds), connect with the flood sub-simulator, share with it the initial topology of the world, and add connecting peers. Once a simulation cycle terminates, the controller updates the time-step and starts the next cycle. Note that, due to the modularity of the above architecture, it is reasonably easy to add as many disaster sub-simulators (e.g., landslides, earthquake, etc.) as needed.

Flood sub-simulator simulates a flood in Trento town (Italy). The equation defined in its core OKC is based on flooding levels and flooding timings deduced from existing flood models. The flood sub-simulator has been developed in Java and it is fully integrated into the OK platform. The main component is an OK peer that subscribes to two IMs: the first is enacted at the beginning of the simulation to share the topology (e.g., point of interests and roads) of the world between the controller and the flood sub-simulator peers and to store, in the controller local knowledge, the connection state of the sub-simulator; the second one, ("Flood", in Figure 3) is used by the controller at every time-step, to get from the flood sub-simulator the water level changes registered at the topology nodes.

Visualiser enables the GUI used to visualise the simulation. In particular, the GUI shows the information provided by the controller through the "Visualiser" IM. At every time-step, the visualiser receives the changes and updates its history according to the new information. The update results in a change on the GUI. Figure 5 shows the appearance of the GUI in the simulated centralized and decentralized scenarios. A green dot represents a reporter peer available for giving information on the water level registered at its location; a grey dot represents

this peer actually giving this information; the water level at a location is depicted as a blue circle whose size depends on how high the water level is; finally, the hat represents the emergency subordinate.

The lack of space prevents us to describe all the developed LCC IMs. For more details, we direct the interested reader to the technical report [3].

5 THE EXPERIMENTAL TESTBED

This section describes the evaluation of the OK framework in the e-Response domain. A series of experiments simulating both centralized and decentralized information gathering strategies were conducted with a three-fold aim:

1. Show the OK system in action, illustrating that all parts of the system are capable of working cohesively in the desired manner.
2. Verify that the technology provided by OK supports different information-gathering patterns.
3. Establish whether the OK paradigm can make positive differences in performance between such disaster scenarios. In particular, what is expected (and desirable) is to have the OK P2P framework comparable or even improving traditional centralized systems when specific fault conditions arise.

In respect to the last point, the hypotheses to be tested are that:

1. under ideal conditions, the OK system exploited in its decentralized nature is comparable in performance to traditional centralized systems and
2. it improves on conventional centralized systems when specific fault conditions arise. In this paper, we only test the first hypothesis, leaving the exploration of the second one to future work.

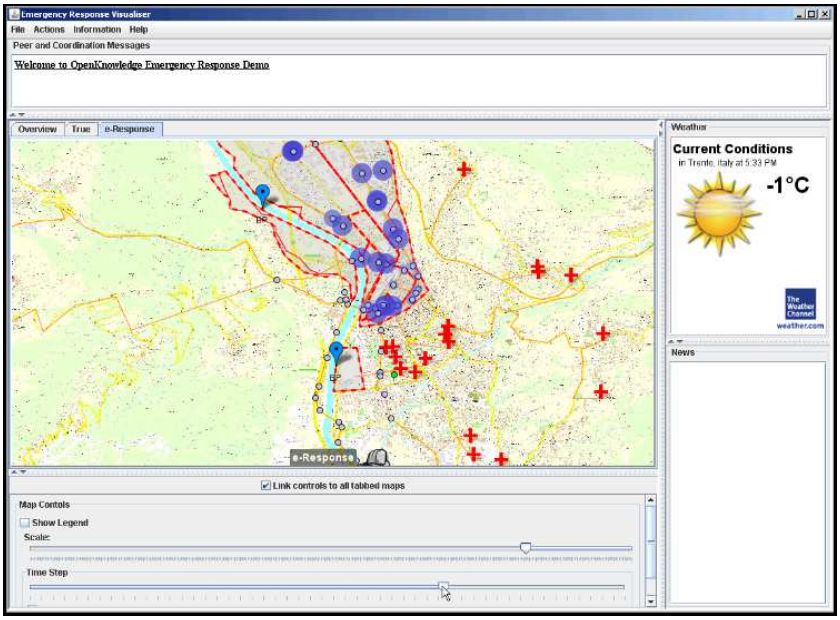
5.1 Experimental Design

To design our experiments we established adequate performance measures, analyzed the variables involved, and set some assumptions.

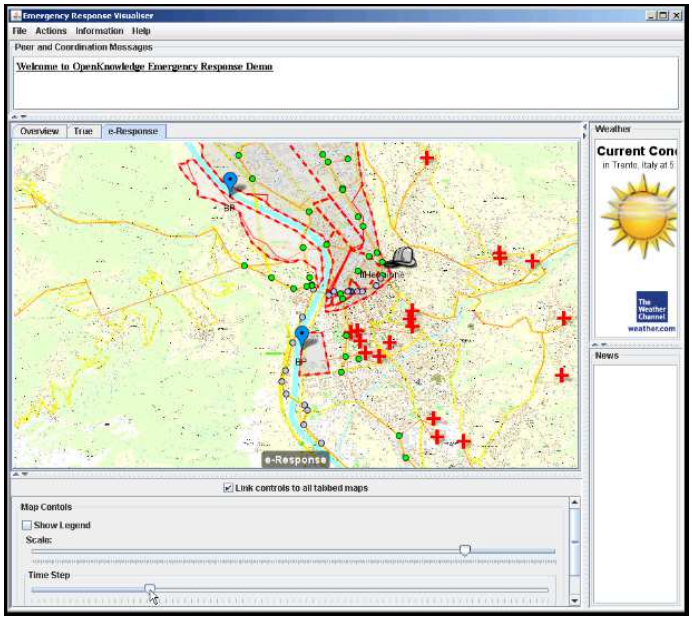
Performance measures for the presented experiments:

1. the *percentage of times* an emergency subordinate arrives at destination;
2. the *travelling time*, i.e., the number of time-steps needed to reach the destination.

These indicators are used to compute and compare the experimental results.



a)



a)

Fig. 5. e-Response GUI; a) Centralized and b) Decentralized information-gathering

Experimental variables – a list of the variables considered follows:

A. *Number of moving peers*: the number of peers moving to a specific destination. Since the main aim is to compare two different strategies (centralized vs decentralized) rather than to make a realistic simulation, this variable is fixed to 1 in all experiments. By running an experiment a certain number of times, we compute the performance 1. of the simulated scenario.
B. *Distance*: set of paths in the topology connecting two locations. To get significant results it is crucial to consider, for each experiment type, the routes covering both safe and flood-prone areas.
C. *Flooding law*: how the flood evolves over time. The flooding law, which is fixed in our experiments, strongly affects the outcome of an experiment run: a peer may either arrive at destination or be blocked depending on how rapidly the flood propagates along the route taken.
D. *Number of nodes*: locations included in the topology and those whose status can be reported by some peer. Incrementing this number is useful to test the capacity of the OK kernel to support many peers. In our testbed, this variable is the number of nodes composing only those routes involved in a given experiment.
E. *Number of (reporter) peers per node*: the number of reporters located in one node. As before, this variable is useful to test the robustness of the OK kernel and, moreover, the effectiveness of some of its modules.

Assumptions – in order to interpret the results in a reasonable way, the following assumptions were set:

1. the CP peer has infinite resources (under ideal conditions). This means that the peer is able to serve any number of simultaneous requests and the communication channel never breaks; therefore, under this assumption, bottleneck problems due to overwhelming requests never occur;
2. a querier, asking a certain number of reporters for an info, will receive all the answers within a time-step. This is due to how the time-step interval is set: the value is such that the time elapsing between one time-step and the next one is sufficiently high to guarantee the replies from all the reporters.

With such assumptions, we simulate a scenario where advantages and disadvantages of both centralized and decentralized architectures are kind of balanced. The configuration of the experiment is sketched in Table 1: each experiment is run 10 times; at each run, the only variables that change are the destination assigned and the locations where reporters are present. Such locations are determined according to the set of routes associated with the destination assigned. The flooding law, the number of emergency subordinates and reporters remain unchanged during all runs.

Each experiment consists in the simulation of the evacuation scenario described in the previous section. Independently on the kind of strategy adopted, the final

			Variable Settings				
Exp N°	**Information Gathering**	**Runs**	**A**	**B**	**C**	**D**	**E**
1	centralized	10	1	1 distance/run	fixed	70 x run	1
2	decentralized	10	1	1 distance/run	fixed	70 x run	1

Table 1. Experiments configuration (no fault conditions)

goal of an emergency subordinate is to safely reach the assigned destination. There are three possible experimental *outcomes*:

1. the agent reaches the destination by following the first route found;
2. the agent finds blocked routes but finally reaches the destination after a number of alternative paths and
3. the agent does not reach the destination at all.

5.2 Experimental Results

After having run an experiment, the relative simulation was visualized on the GUI in order to analyze the movements of the emergency peers and verify the correct mechanism in the coordination among the agents.

Figures 5,a) and b) show a simulation run for the centralized and decentralized scenario, respectively. Figure 5 a) shows the agent out from the flooded area. Here, all the dots are grey, meaning that all reporters are being queried by the CP in order to obtain the water level of their location. Some of them register high levels of water. Figure 5 b) shows the agent moving along a route which can be deduced by the grey dots ahead of the agent; in fact, these dots represent those reporters located along the route followed and therefore queried by the moving agent; all the other reporters remain available (green dots). Here, the OK paradigm is exploited in its decentralized nature, since the information-gathering is based on the use of distributed information reporter agents and not on a unique provider, as in the first case.

Figure 6 a) shows the outcome distribution obtained by running 10 times the first experiment. As can be seen, 70 percent of the time the experiment has outcome 1) (the peer reaches the destination without problems) while 30 percent of the time the outcome is 3) (the peer does not reach the destination). The outcome 2) is never obtained. Although we setup the routes in order to cover different kind of areas (either safe or flood-prone areas), the case where an agent finds free routes after a re-routing never happens. This could be explained by considering how the design of the flooding law and its related "flood speed" affect the evolution of the scenario. The outcome distribution related to the second experiment, which simulates the decentralized scenario, is identical to the one found for the first experiment and hence is not reported here. This result can be explained with the assumptions previously made: asking information on the route's practicability to either the CP or reporters scattered around the city does not make the difference.

Figure 6 b) shows the time taken (measured as the number of simulation time-steps) by an agent to reach the goal location according to the shortest distance (in terms of intermediate locations) between the initial position and the final destination. The trend is shown for both experiments. It can be observed that, in both cases, the time needed to achieve the goal is nearly equal to the shortest distance. This can be explained by how the simulation is designed – an agent moves from a location to the next one exactly in a time-step – and by the missing outcome 2). Finally, Figure 6 b) reveals very similar trends for both centralized and decentralized scenarios. Again, this is mainly due to the assumptions made and the variable settings.

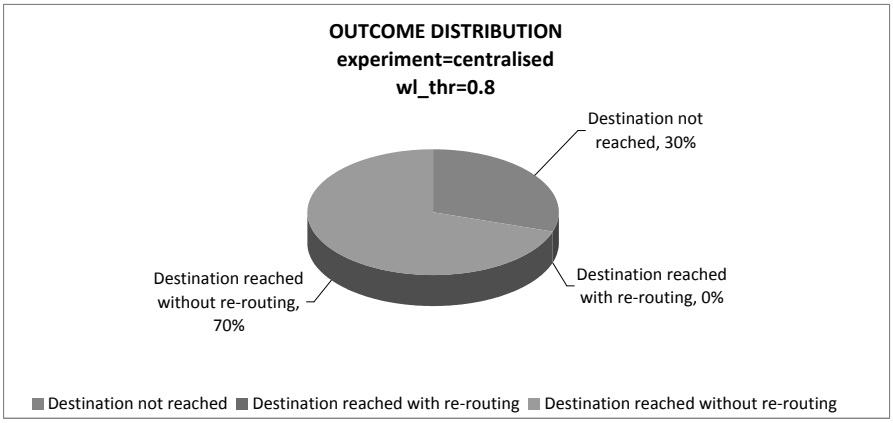
In view of the results described above, we can conclude that our first expectation is met: the use of the OK framework supports both architectures (centralized and decentralized) and provides comparable performances under the selected – ideal – assumptions.

6 CONCLUSIONS AND FUTURE WORK

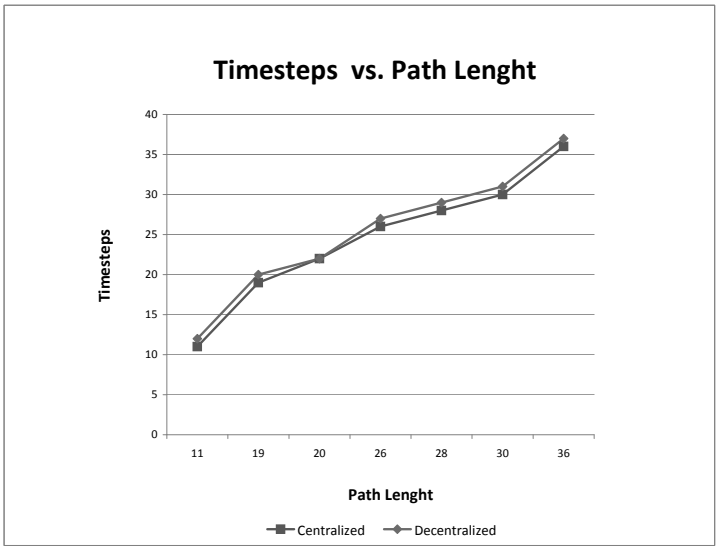
This work focuses on testing whether the OK framework is capable to support the coordination of emergency activities and how, in absence of fault conditions, the OK p2p framework is comparable in performance to traditional centralized gathering approaches. An agent-based e-Response simulation system fully integrated with the OK infrastructure has been developed. It is used to model specific emergency scenarios and agents in terms of both LCC specifications and OKC components. A suite of experiments has been designed and run to evaluate the performance of the OK e-Response system in different scenarios and under specific assumptions. The preliminary results thus obtained show how the OK infrastructure is equally effective in both centralized and decentralized information-gathering.

We are currently working on further experiments. In particular, we want to repeat the described experiments by increasing the number of runs and by tuning parameters like the "flood speed" and the routes to follow, so that different outcomes can be obtained. This way, we could reconfirm our hypothesis in a more robust setting. Also, we want to run experiments where two types of fault conditions are injected: failures in the communication channels and inaccurate signaling. For this, the following variables will be considered:

- *Distribution of trustworthy (reporter) peers*: number of trustworthy reporters, i.e., peers always reporting accurate water level values. By setting this variable, the fault due to inaccurate signaling, its location and its severity can be simulated.
- *Degradation of the CP communication channel*: measured as the likelihood of a fault in the communication channel of the CP agent. For example, a degradation of the 80 % means to have this agent serving incoming requests only 20 % of the times.



a)



b)

Fig. 6. a) Outcome Distribution, b) Time-steps vs. Path Length

- *Distribution of degraded reporter's communication channels*: defines, for each reporter communication channel, the probability of its disruption.

With these new experiments, we can investigate if – and eventually under which conditions – a complete p2p architecture improves the overall performance and robustness over traditional centralized architectures. Finally, in respect to the simulated scenarios and the involved agents, it is worth to consider the reporter agents as mobile agents rather than fixed sensors, and thus to explore how

the OK platform supports the coordination of team-members in an emergency site.

Acknowledgements

This work was supported by the OpenKnowledge project (FP6-027253).

REFERENCES

[1] Siebes, R.—Dupplaw, D.—Kotoulas, S.—Perreau de Pinninck, A.—van Harmelen, F.—Robertson, D.: The OpenKnowledge System: An Interaction-Centered Approach to Knowledge Sharing. In: CoopIS 2007.

[2] Robertson, D.: A Lightweight Coordination Calculus for Agent Systems. In: Lecture Notes in Computer Science – DALT, Vol. 3476, 2005, pp. 183-197.

[3] Trecarichi, G.—Rizzi, V.—Vaccari, L.—Pane, J.—Marchese, M.: Openknowledge Deliverable 6.8: Summative Report on Use of OK Approach in eResponse: Integration and Evaluation Results. Technical report, Openknowledge project, 2008.

[4] Marchese, M.—Vaccari, L.—Trecarichi, G.—Osman, N.—McNeill, F.—Besana, P.: An Interaction-Centric Approach to Support Peer Coordination in Distributed Emergency Response Management. In: IDT, Special Issue on Incident Management, Vol. 3, No. 1, 2009, pp. 19–34.

[5] D'Aprano, F.—de Leoni, M.—Mecella, M.: Emulating Mobile Ad-Hoc Networks of Hand-Held Devices. The OCTOPUS Virtual Environment. In: Proc. of the ACM Workshop on System Evaluation for Mobile Platform: Metrics, Methods, Tools and Platforms (MobiEval) at Mobisys, 2007.

[6] Mecella, M.—Catarci, T.—Angelaccio, M.—Buttazzi, B.—Krek, A.—Dustdar, S.—Vetere, G.: Workpad: An Adaptive Peer-To-Peer Software Infrastructure for Supporting Collaborative Work of Human Operators in Emergency/Disaster Scenarios. In: CTS 2006.

[7] Bellamine-Ben, Saoud N.—Ben Mena, T.—Dugdale, J.—Pavard, B.—Ben Ahmed, M.: Assessing Large Scale Emergency Rescue Plans: An Agent Based Approach. Special Issue on Emergency Management Systems. In: IJICS, Vol. 11, 2006, No. 4, pp. 260–271.

[8] Kanno, T.—Morimoto, Y.—Furuta, K.: A Distributed Multi-Agent Simulation System for the Assessment of Disaster Management Systems. In: IJRAM, Vol. 6, 2006, No. 4–5, pp. 528–544.

[9] Massaguer, D.—Balasubramanian, V.—Mehrotra, S.—Venkatasubramanian, N.: Multi-Agent Simulation of Disaster Response. In: ATDM Workshop in AAMAS 2006.

Gaia Trecarichi is a Ph. D. student at the University of Trento. She is currently working on the "LiveMemories" project for the development of an advanced Web 2.0 platform enabling communities in the construction of personal and collective memories. She is also involved in the "Glocal" project, whose aim is to build an event-based multimedia retrieval system. She worked on the "OpenKnowledge" project and, in particular, on the evaluation of the OK framework in the e-Response domain. She obtained a Master Degree in computer engineering from the University of Palermo.

Veronica Rizzi is a research technician at University of Trento. She currently works on the "LiveMemories" and the EU-funded "Glocal" projects. She holds a postgraduate degree in computer science from the University of Trento.

Maurizio Marchese is Associate Professor of computer science at the Department of Information Engineering and Computer Science, University of Trento. He is author of over 80 publications and has been program coordinator of a number European Research Projects. His main research interest are: the design and development of service architectures in distributed systems; the integration of services in Geographical Information Systems (GIS) environments; the analysis, development and integration of services to support scientific knowledge creation and dissemination processes.

Lorenzino Vaccari has received his Ph. D. examining the semantic interoperability between geographic web services from the University of Trento, Italy; his M. Sc. from the same university, and his B. Eng. from the Turin Polytechnic, Italy. He is now working at the European Commission (DG JRC) as a post-doc researcher. Working in the ENABLE action of the SDI Unit, he undertakes research to support the development of the European Spatial Data Infrastructure foreseen by the INSPIRE directive, focusing on the interoperability of spatial data and services. He previously worked in the FP6 EU-funded OpenKnowledge project that investigated the dynamic and semantic integration of components and services in open, peer-to-peer (P2P) systems. Prior to joining the JRC, he worked for thirteen years at the Autonomous Province of Trento (PAT), as a GIS/SDI specialist. During his permanence at PAT, he also joined the Italian geographic national committees of CPSG and SINA. His previous work has also included software design in the field of real time embedded systems, closed loop systems and PID controllers in distributed and centralized environments. His research interests include semantic interoperability and spatial data infrastructures, GIS management, P2P architectures for emergency response and semantic web issues.

Paolo Besana is a post doctoral researcher at the Université de Rennes I. He is currently working on "ASTEC" project for automating the prescreening of patients for clinical trials. Previously he worked in the UK-funded "Safe and Sound project", aiming at creating a distributed approach to medical decision support systems and in the EU-funded OpenKnowledge project, whose target was the dynamic integration of the components and services in open, peer-to-peer systems. He holds a Ph. D. in Informatics from the University of Edinburgh and a postgraduate degree in telecommunication engineering from the Politecnico of Milan.