A location-based approach for distributed world-knowledge in mobile ad-hoc networks

L.J.M.Rothkrantz, M. van Velden, D. Datcu

Abstract: *In this paper we consider a building-like environment during crisis context where individuals explore an unknown world. It is assumed that each individual is equipped with a PDA which is wireless connected (MANET). Users can enter information about their position in the dynamically changing world. The main focus is on automatically building a map of the world by using observations from individuals in such an infrastructure-less network. We have designed and implemented this proof of concept, ManetLoc, in the form of a simulation.*

Key words: *mobile ad-hoc network, agent, communication, topological map, infrastructure-less, location, emergency, awareness, crisis.*

INTRODUCTION

In every aspect of our lives we are becoming more and more dependent on the availability of (information) systems. In times of crisis not all of these systems might always be readily available. If a telecommunications infrastructure is not available at a certain area, it should still be possible to set up an infrastructure-less network or mobile ad-hoc network (MANET) under most conditions. Setting up such a network enables us to share information concerning the state of the world and coordinate actions. These ad-hoc networking technologies are making it possible to exchange information anywhere, anytime without prior network infrastructure. Using handheld devices that operate in a wireless environment, communication is still possible when major infrastructural communication links have been damaged, destroyed or overloaded. So in case of a major disaster within a city, emergency services can communicate without the need for preset-up access points or other such infrastructural requirements.

At Delft University of Technology and DECIS lab, there is a research project focused on crisis management. The current paper describes the results of a sub-project that aims at designing and implementing a multi-agent-system that can operate in environments without a pre-setup infrastructure (only a mobile ad-hoc network) and without any pre-knowledge of the world. The system is able to process and fuse location information from different users and sensors remote in space and time and to distribute this information through the network.

RELATED WORK

In this section some related topics will be described. The first topic that is important for our work is that of agent systems, and more specific, multi-agent systems (MAS) [1]. Many MAS systems are developed for wired networks, but only a few for wireless environments.

Mobile Ad-hoc Networks (MANETs) are wireless networks consisting entirely of mobile nodes that communicate on the move without base stations. Nodes in these networks will both generate user and application traffic and carry out network control and routing protocols. MANETs are very flexible because of the dynamic topology where nodes are free to move arbitrarily and of the fact it allows Peer-To- Peer (P2P) communication in an asynchronous manner.

Small devices are portable computing devices with networking capabilities, such as a mobile phone or a PDA. Limited battery life and connectivity are the current most relevant issues in our project. As we are not depending on infrastructure but on ad-hoc network technologies the latter is this most constraining issue. Nowadays it is possible to execute relatively complex software applications such as route planners on small devices and it is to be expected that these devices will become more and more powerful in the near future.

The research reported here is part of the Interactive Collaborative Information Systems (ICIS) project, supported by the Dutch Ministry of Economic Affairs, grant nr: BSIK03024.

The device we take into account is the Sharp Zaurus SL-C760. This PDA has a Linux based operating system, Qtopia, X, Java and Jade installed.

Location awareness plays a crucial role for context aware agents. From this point we focus on the possibilities for the determination of the position of nodes in mobile ad-hoc networks. If a system like GPS is available [8], each node can be easily aware of its own location, but if GPS is not available (for some or all nodes) relative positions have to be determined using other methods. There are systems and system concepts available, which use network parameters such as time of arrival to calculate positions [7] [4] [9].

MODEL

We developed a simulation for MANET system. In the simulation system, maps of an environment can be created and distributed. The main goal of our project involves storing and distributing of location dependent context information based on user observations. The processes involved are primarily based on the positions different virtual people visit and any context information they might gather, such as where exits are and determining the shortest route to one. In our program each node will never know its exact world coordinates and initially does not have any knowledge of the world i.e. there is no knowledge of absolute position and no initial map of the world. The goal for each agent-node is to construct this internal map of the world by using its own observations and sharing information with other nodes. In this way, the knowledge of one node is distributed to the other, nearby nodes. In practice this means that nodes are within communication range. In our case the data shared relate to information about halls and crossings and can be extended to suit the full usage needs. The basic modules of the system are:

- *Gathering data*

Before anything else, methods for gathering data on location must be in place. The collected data can then be transformed into knowledge and used for constructing and maintaining a world model. In our system data about the world is gathered in 3 ways:
Sensory data: the most trivial is that the system keeps track of the distances the user travels. For this it could count the number of steps the user makes and based on that (and possibly other data) to make estimations on the covered distances.
User input: preferably by voice recognition users supply information to the system, but other methods such as a point and click system could be used as an alternative or addition to the projected use of voice recognition.
Other agents: information provided by other agents within communication range can be merged with the agent's own world model.

- *Building topological maps from user and sensory input*

The first goal for our agent-system is to be able to construct a local world model from data the user and sensors provide about the world. The most trivial input a user can provide for building a map of a building is to indicate that at a certain time he encounters an intersection and possibly to indicate the number of paths and their directions. A graph created on observations and actions from user reports can be displayed on the screen of the user's PDA. To be able to generate such a map it is important to be able to estimate (relative) distances traveled. These distances can be calculated from the time between two observations, combined with the type of movement (running, walking, etc) and can be refined by using pedometer input. There are more possibilities but we prefer to use as little specialized hardware as possible. In our system we will make the assumption the user provides relatively accurate indications of distances traveled (+/- 10%). The used algorithm will be described in the next section.

- *Sharing and merging location context information between agents*

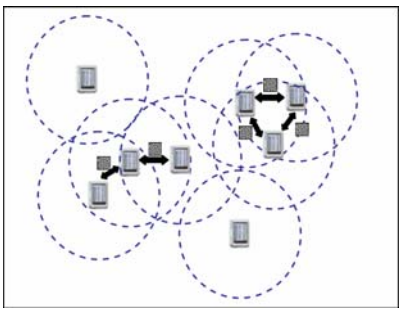

Fig. 1 Agents in MANETs sharing maps

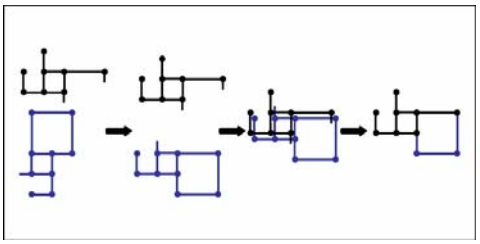

Fig. 2 Map merging process

The agents are operating in a mobile ad-hoc networking environment in which it is likely they encounter other agents every now and then. When an agent detects one or more (compatible) agent(s) within its communication range, they share (figure 1) and possibly merge (figure 2) the knowledge they have about the world [3] [5] [6]. Each agent will attempt to merge information from other agents for themselves. Consequently it is possible for two agents to come to different conclusions. This makes it possible for PDAs to have different agent software (versions) running or to be in a different processing modes. The only requirement is that the messages they send out must be compatible.

Similar to the internally stored world models, maps are shared in the form of graphs. They are assumed to be consistent, but not exact and do not always have a common reference frame. If two or more users have explored overlapping regions of an environment their agents should have topological maps that have common sub graphs [11] [2] with identical structure. Since having a common sub graph it is possible to find a reference frame and merge the two maps into one.

In order to be able to merge two maps, we first need to match the maps together, building hypothesis and choosing the correct one (i.e. the best match). A hypothesis is a possibly rotated sub graph that the two maps have in common. There is a chance the process described below does not supply a (large enough) hypothesis. If this is the case and not enough vertices can be matched to make a good hypothesis yet, the map received is stored internally. In that case we can try the matching process later on, when a more complete world model is available. The used algorithm for matching two maps consists of three phases, as described in the next section.

- *Providing services based on location context information*

When an agent running on a user's PDA gathers knowledge about the world, the knowledge can be used to provide services such as advice to the user. One of the possibilities is guidance of the user. In the context of the crisis management project, knowledge could be gathered about crisis indicators (fire, smoke, etc) and translated into some sort of scenario (e.g. the building is on fire, south side is still clear to pass). It is not unthinkable that having this type of knowledge available in an agent network could even be used to coordinate the actions of individuals and groups.

ALGORITHMS

Closing the loop

If a user would travel long enough in a building he will eventually always return to a location that was visited before from a different direction. The process described in the example will then result in a loop in the graph, which should be detected and closed. A correctly detected closed loop is very valuable information, as it is required to make a map consistent. Consistent maps are required by our system to be able to match and merge

one map with another. The question is how to detect such a closed loop [10]. As an example of closing loops we take an even simpler square shaped world. Applied to this square world the process described above might result in the following 'map', where the two upper-left intersections, should be recognized as one and the same, but at the moment we still have 5 vertices in the graph where there are only 4 intersections in the world (figure 3).

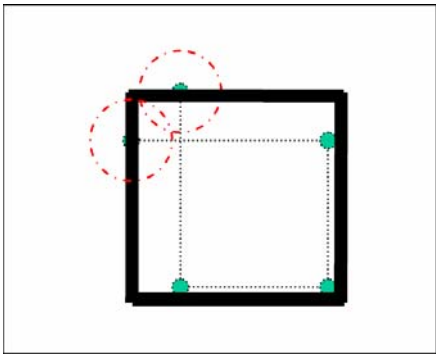

Fig. 3 Open loop

Before being able to close a loop we should be able to detect and build hypotheses concerning possible loops. For this we roughly follow a procedure that checks for the existence of vertexes next to a given one that might close a certain loop. If we can find two matching vertexes, then a loop hypothesis is started. To make the graph consistent when two matching vertexes are found, small adjustments can be made to the endpoints of edges. Whenever a loop hypothesis has been formed we can start testing it by comparing edge lengths of the supposed loop with new measurements. These measurements will result in accepting or rejecting the loop.

Matching maps

The used algorithm for matching two maps consists of three phases:

- *Vertex matching*
- *Growing hypotheses*
- *Combining hypotheses*

Vertex matching

The first step taken into account for map matching is building a list of all vertices that match each other in the two maps. Two vertices only match if they have the same edge directions. This is also the case if a vertex needs to be rotated to match, which is also stored. We expect exactly known attribute vertices, such as the type of the vertices to match perfectly. However, attributes that are subject to measurement error can be compared with a similarity test. In the case of our simulation we don't have any fuzzy variables of a vertex, but in a real environment or an extension of our simulation such variables might occur.

Growing hypotheses

As soon as the list of matching vertices is available, we analyze the matches by testing corresponding pairs of edges and leaving the paired vertices. If the edges are compatible and the vertices at the ends are also compatible, they are added to the hypothesis. If the edges or vertices are incompatible, the entire hypothesis is rejected. The vertices are tested with the same type of criteria and similarity tests used to form the initial pair. Edges may also have both exactly and inexactly known attributes. In our system, they have their path length compared with a similarity test. Our hypotheses are the unique matches

surviving the growing process. Duplicate hypotheses are avoided by keeping a table of vertex pais. When vertices are paired during the growth phase, the corresponding entry is marked in the table. This entry corresponds to an initial pairing of vertices. A sub graph in one map can be matched to multiple sub graphs in the other under separate hypotheses, but a pair of matched vertices with a given edge correspondence can appear in only one hypothesis. The matching and growing process is repeated until all valid vertex pairings are examined. Please note that if we would be working with imperfect user input - which is not the case in this concept – a procedure filtering noise should be added. When able to match enough common vertices and edges, and if there is a minimal amount of conflicting vertices and edges, the conflicts can be discarded and the hypothesis accepted.

Combining hypotheses

If successful, the hypothesis growing process described above results in a list of possibly multiple hypotheses within the same rotation. From these hypotheses one has to be selected. Before this takes place it is possible that if such a list contains more than one hypothesis, some of these entries are consistent with each other. These hypotheses are then combined with each other into one larger hypothesis cluster. After the system has chosen a hypothesis cluster, the next step is to merge (or flatten) the two maps into one single map. Estimates of path lengths can be updated by combining the measurements from the two maps for corresponding edges. The edge orientations at the corresponding vertices can be similarly merged. The parts of a given map that are missing should be added. The merging process is globally performed in four steps:

- *Rotate the received map so its orientation matches the local node's map*
- *Shift the rotated map so its coordinates match the local agent's map*
- *Add any new vertexes from the rotated and shifted map to the local node's map*
- *Connect everything together (update edge lengths, check for inconsistency's etc.)*

System Tests

The simulation was initially performed on a 10 and 30 nodes map, first with just one agent exploring the world automatically using nearest unexplored area navigation, and later with more agents who were simultaneously started. The time it took an agent to find the complete map was measured and also was always checked if the output was correct.

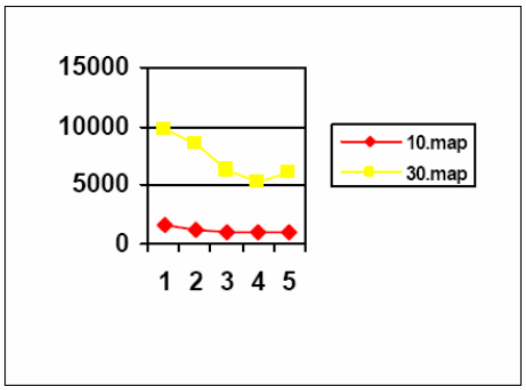

Fig. 4

Table 1

Map	#agents	Avg. units
10.map	1	1721
10.map	2	1125
10.map	3	1100
10.map	4	1112
10.map	5	1060
30.map	1	9701
30.map	2	8569
30.map	3	6227
30.map	4	5301
30.map	5	6017

The results (figure 4, table 1) clearly show that the process of sharing and merging maps has an effect, the larger the map the larger the effect and also the more agents the more gained. Agents start-in in a map that was already explored by others logically have the most gain; after they have explored a small part of the world they can simply merge the large map parts with their own. Please note that the test result would probably be

significantly lower if the Ad-Hoc Simulation (AHS) network code wouldn't suffer from deadlock issues. In another test the 30 intersections world was pre-explored completely by 5 agents and after that a fresh agent was added. The new agent received the complete map from multiple agents and was able to find the complete correct map within 536 distance units traveled. Considering it takes 9701 units on average for an agent to explore this world on its own this is a considerable gain (in this specific case 18 times faster).

CONCLUSIONS

Our experiments conducted for a simulated world show that it is very well possible to distribute, and merge world knowledge in a mobile ad-hoc multi-agent environment. Even in such an environment with limited communication possibilities our test results showed there is a significant gain found when solving a mapping problem with multiple distributed agents. As expected, the larger the map the better results on how useful distributing and merging partial maps is. Although with the fact that simulation runs on a single machine comes that there are limits because of processing power. This is an important issue still encountered in our simulation system. Though when calculations would not be performed on one machine anymore, but on one for each agent, it should not be a problem anymore. So we anticipate scalability will not be a direct problem, should the simulation be translated into a real life distributed system.

REFERENCES

[1] Agents. The American Association for Artificial Intelligence, 2000 – 2005.

[2] H. Bunke, A. Kandel, Mean and maximum common subgraph of two graphs, University of Bern, University of South Florida, 2000.

[3] Z.Butler, A Rizzi, R. Hollis, Distributed coverage of rectilinear environments, Carnegie Mellon University, 2001.

[4] S. Capkun, M. Hamdi, J. P. Hubaux, GPS-free positioning in mobile Ad-Hoc networks, Ecole Polytechnique Federale de Lausanne, 2001.

[5] G. Dedeoglu,G. Sukhatme, Landmark-based matching algorithm for cooperative mapping by autonomous robots, University of Southern California, 2000.

[6] K. Doty, S. Seed, Autonomous agent map construction in unknown enclosed environments, University of Florida, 1994.

[7] T. Kitasuka, T. Nakanishi, A. Fukuda, Design of WiPS: WLAN-Based Indoor Positioning System, Kyushu University, 2003.

[8] L. D. Murphy, T. Murphy, Global Positioning Systems, A Technical Assessment Paper, 1997.

[9] D. Niculescu, B. Nath, Ad-hoc Positioning System (APS) Using AOA, Rutgers University.

[10] F. Savelli, B. Kuipers, Loop-Closing and Planarity in Topological Map-Building, Universit`a di Roma "La Sapienza", University of Texas at Austin, 2004.

[11] G. Valiente, Subgraph Isomorphism and Related Problems, Technical University of Catalonia, 2001.

ABOUT THE AUTHOR

Assoc. Prof. L. J. M. Rothkrantz, Department of Man-Machine Interaction, Delft University of Technology, Phone: +31 15 2787504, E-mail: L.J.M.Rothkrantz@ewi.tudelft.nl.