

# Arian Rescue Simulation

Members:

- Mazda Ahmadi
- Ali Nouri

0-Preface .....

1-Technical appeal of the team.....

1.1 Powerful inter-agent communication.....

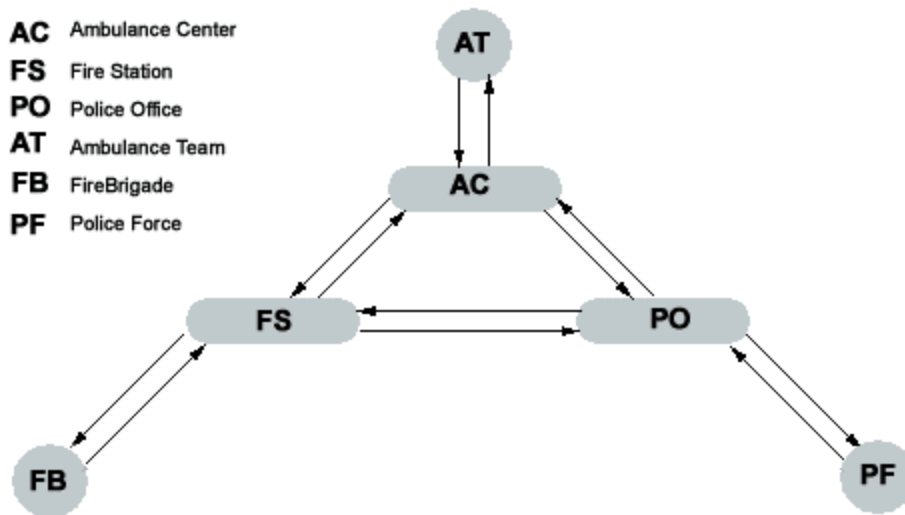
1.2 reductions of classic algorithms.....

## 0 Preface

At March 2001 our team started with 4 members. After working for a month we decided to continue with 2 members, Ali Nouri and Mazda Ahmadi. We are both undergraduate juniors and close friends. We found Rescue Simulation interesting because of different reasons. First the fact that it is going to do some good for mankind second we liked multi agent programming.

## 1.1 Powerful inter-agent communication system:

Among all the other fields of Robocup, with 6 different type of agents and very limited eyesight, rescue simulation has the heaviest communication load in the network.so use of a good communication manager is inevitable in the system. An overview of our whole communication system is shown in the figure 2.1.



Because of the limit we have on the number of messages and also the delay in transmitting message between agents, the communication system can not be bi-directional though some of the

messages are lost due to the limit of 4 message rule. The most common communication paths in the figure above are:

- FB/PF/AT to AT used when some agents are buried in the building
- FB/AT to PF used when some agents are trapped in a dead-end
- FB/PF to AT used to report buried civilians.
- AT to AT used to inform other partners about our decisions, etc.

The main problem for communication between agents here is that all the mentioned communications are multi hopped. For example, there is only one-way from FB to AT in the above graph and that is: FB-FS-AC-AT. The hardness of this kind of communication in our system can be described as follows:

- Once a member of the path in the graph cannot hand in the message to the next one, the whole message is lost.
- 90% of the messages are delivered very late. It takes more than 6 cycles for the messages to reach their destination.

To overcome these problems we have figured out some tricks to reduce the network traffic and to safen the whole system. Some of them are:

- All senders will keep track of sent and received messages in order to prevent overdoing.
- Some of the paths can be reduced as follows: A path of FB-FS-AC-AT can be reduce to B-AT and a AT-AT piggybacked message, if there is a AT agent near FB agent so that the message can be sent via a "say" command instead of "tell" command.
- Fuzzy logic can greatly improve the system and prevent the system from a lot of messages. E.g. If the PF agent be, somehow, able to have a good fuzzy view of the world (about the crowded areas, fire places, etc.) then it can dynamically change it's current algorithms for the new ones.
- Single messages can themselves be mixed. With use of a message splitter manager, we can embed more than one kind of message in a single one.

And there are a lot more about inter-communication manager in a multi agent system, which we took care of.

## 1.2 Reduction of classic algorithms:

Once the information about the world's current state has been successfully passed to each other, the agents will enter the "Decision Making" state. In this state, we're faced with this question: "what is the best target?" to answer this question we have to take care of two significant points about our problem:

- Picking the best target in our system is highly dependant on the other agents' actions. So the algorithms we choose for our agents should fit each other's.
- Our system is a little bit like "Real time systems". That is late results are worse than bad results. Because of the behavior of the kernel, the action of the agents should be transmitted back to kernel at the end of the current cycle and latecomers are discarded.

The best solutions, especially in graph related problems, are usually derived from classic algorithms. A Floyd algorithm can give the shortest path between every two nodes in a graph, Of course if we have enough time and information. But the problem here is that we lack both of the items. Most of the classic algorithms are highly time consuming and cannot be easily used in our systems. Also lack of information about the whole world's state prevents us from applying simple classic algorithms.

For example in FB, we wanted to gain the connected components of a huge graph with a lot of edges (near  $n$  power 2). We tried some classic algorithms such as DFS and BFS but we soon

found that the agents miss to send the information to kernel on time and they loose some of the cycles. So we tried to find some new methods for the problem and we came up with a very tiny method to do the job and it really worked in more than 80% of the time. The idea was simply taken from some salutes usually done in the opening of a sport event, where lines of people standing side by side form a shape. Since the graph was based upon the location of the objects in the city, we applied the same trick for getting the connected components of the graph. There were several cases where we replaced heavy -duty classic algorithms by simpler ones.