

Agent Management Architecture for a Large-scale Earthquake Disaster Mitigation Simulation System

Tetsuhiko Koto¹ Ikuo Takeuchi^{1,2} Itsuki Noda³ Nobuhiro Ito⁴
Hiroki Shimora² Takuya Morisita² Ken'ichi Honji⁴
Tomoichi Takahashi⁵ Yoshitaka Kuwata⁶

¹The University of Electro-Communications

²Kawasaki Laboratory Earthquake Disaster Mitigation Research Center
National Research Institute for Earth Science and Disaster Prevention

³National Institute of Advanced Industrial Science and Technology

⁴Nagoya Institute of Technology

⁵Meijo University

⁶NTT DATA CORPORATION

Abstract

Integrated disaster simulation systems are important tools to decrease the damage caused by various disasters. The paper describes the agent management architecture for the IDSS (Integrated earthquake Disaster Simulation System). The IDSS is a distributed simulation system which integrates various sub-simulators and various agent programs, or simply agents. We propose an agent proxy sub-simulator (APX), which provides a suitable API for agents.

1 Introduction

Integrated disaster simulation systems are important tools to decrease the damage caused by various disasters. In peacetime, it would be used for disaster-proof city planning, disaster prevention plans, and disaster response training. At the time of a disaster, it will be used for real-time simulation to predict disasters and help decision making.

The paper describes the agent management architecture for the IDSS [1, 2] (Integrated earthquake Disaster Simulation System). The IDSS is a distributed simulation system which integrates various sub-simulators such as a seismic intensity simulator, fire and extinguishment simulators, rescue activity simulators, agent health simulators, traffic simulators, and so on. The IDSS also simulates a

huge number of various agents such as civilians, fire fighters, members of rescue parties, and so on. We designed the architecture and the interface for sub-simulators [2, 3], but it is not suitable to treat individual agents as individual sub-simulators, because the number of agents is huge, and most of them need only very simple computation for their action decision.

2 IDSS Architecture

The sub-simulators are classified into two categories; those for (1) disaster analysis and estimation, and (2) disaster prediction and response. Sub-simulators for disaster analysis and estimation calculate the initial environment for agent activity simulation, and sub-simulators for disaster prediction and response calculate the change of the world reflecting agent activities.

The interaction of sub-simulators for disaster prediction and response is implemented using shared *space-time graphs*. The space-time graph is an abstract data structure representing a table, whose horizontal axis and vertical axis represent time and state variables, respectively. A row of the graph represents the time series of the values of a state variable. A column represents the snapshot of the simulated world at a point of time. The graph is initially blank, and filled from left to right, step by step, as sub-simulator simulation proceeds. A sub-simulator reads the current situation of simulation from it, and updates it with the result of a simulation step. The states of the other sub-simulators are provided in the space-time graph. If it is filled completely, the simulation comes to an end. The time scale of the graph is millisecond. Sub-simulators, however, write the graph with their preferred time scale.

To simulate disaster over large areas, the simulated world is divided into some number of disjoint *regions*. For each region, there are a space-time graph and all sub-simulators such as a fire simulator and a traffic simulator. One region is assigned to one or more PCs. The edges of each region are overlapped a little with adjacent regions, and the simulation results of the adjacent regions are passed through the space-time graph. A sub-simulator simulates the region represented by its own space-time graph and isn't concerned about the adjacent regions.

3 Problems on Agent Support

An easy way to support agents is to implement an agent as a sub-simulator. However, this approach has the following problems.

Performance An agent accesses the partial data of the space-time graph, that is, an agent reads the information of the surrounding area and writes its activity. If an agent is implemented as a sub-simulator, the system should provide the accessibility to all the space-time graph. This means the system has to send all the space-time graph to each agent.

Movement over regions An agent can move across a region boundary and move to other regions, but the region simulated by a sub-simulator is fixed, so sub-simulators cannot move to another region.

Long-distance communication Agents can communicate each other by telephone, etc. This communication takes place sometimes over regions. However, the area that is accessible by a single sub-simulator is limited to a single region, so new framework to simulate long-distance communication is required.

4 Agent Proxy

We propose the agent proxy sub-simulator (APX). Agents are connected onlt with the APX and the APX bundles the decisions of agents together and sends them in unity to the simulation system as the simulation result of the APX. The APX sends the simulation results of the other sub-simulators to its agents. The APX also migrates agent processes, and manages the long-distance communication of agents such as telephone calls, broadcast, and so on.

For each region, there is one or possibly more APX. It is desirable that an APX and the agents connecting with it are located in a single PC.

4.1 Output of Agents

Every five seconds of simulation time, an agent output *commands* such as *walk*, *drive*, *extinguish* and *say*. The command represents the agent actions for the next five seconds. A command is written into the space-time graph, and the other sub-simulators process the command. For example, the traffic simulator processes walk commands, drive commands and so on to simulate the movement of agents. A simulation step of agents is fixed to five seconds in order to reduce the frequent activation of the simulation system, which brings in much overhead. An agent may issue multiple command at once to indicate taking multiple actions. For example, if an agent walks while talking, the agent sends both the walk command and the say command at once.

4.2 Input of Agents

The input of agents is the situation of the world sensed by the agent. Agents can sense a lot of information, but not all information is processed. For example, the robot that has good sight can see a broad area, but it may watch only the fires. Therefore, it is efficient to provide the proper sensing information on the agent demand, rather than send all information.

Also, we want to simulate various agents that have various sensing abilities such as robots and the human equipped with various devices. It is important to develop these agents easily, so we propose that the sensing abilities of agents are simulated by agents and the APX and the simulation system don't restrict the sensibility.

For above reasons, the APX simply provides the interface to read the space-time graph. An agent specifies a state variable that it requires to read, then the APX returns its value.

4.3 Communication

We are targeting human voice communications and telephone calls currently. Human voices are simply written into the state variable of the agent that describes the sound the agent gives. Telephone calls are processed by the APX. The APX calculates whether the communication is possible, and sends the message to the receiver agent. If the communication crosses over regions, the APX forwards the message to the APX that manages the receiver agent.

4.4 Agent Process Migration

The communication between an agent and the APX is tight, so it is important to migrate the agent process to another PC, if the agent moves to the region managed by the APX on the PC. However, the agent may frequently come and go over two regions. In this case, it is costly to migrate the agent process as soon as the agent moves.

After an agent moves to another region, first, the agent is reconnected to the APX that manages the region. And then, after the agent moves to the inner area of the region or spends some time at the region, the agent process is migrated.

5 Conclusion

We proposed the APX sub-simulator, which bundles the decisions of agents together and sends them to the simulation system, and which sends the simulation results of the other sub-simulators to its agents on demand. The APX also migrates agent processes, and manages the long-distance communication over regions.

References

- [1] Ikuo Takeuchi, Shigeru Kakumoto, and Yozo Goto: Towards an Integrated Earthquake Disaster Simulation System, 1st Int'l WS on Synthetic Simulation and Robotics to Mitigate Earthquake Disaster, July 5, 2003, Padova, Italy.
- [2] Tetsuhiko Koto and Ikuo Takeuchi: A Distributed Disaster Simulation System that Integrates Sub-simulators, 1st Int'l WS on Synthetic Simulation and Robotics to Mitigate Earthquake Disaster, July 5, 2003, Padova, Italy.

- [3] Hiroki Shimora, Itsuki Noda, Yoshitaka Kuwata, Tetsuhiko Koto, and Ikuo Takeuchi: Design of Application Programming Interface for Distributed Simulation System, 1st Int'l WS on Synthetic Simulation and Robotics to Mitigate Earthquake Disaster, July 5, 2003, Padova, Italy.