NAIVE : Network bAsed Intelligent Virtual Environments

N. AVRADINIS, A. BELESIOTIS, I. GIANNAKAS, R. KOUTSIAMANIS, T. PANAYIOTOPOULOS, K. TILELIS
Knowledge Engineering Lab, Department of Informatics
University of Piraeus
80 Karaoli & Dimitriou str., Piraeus 18534
GREECE

Abstract: - In this paper we suggest an approach for developing a network based system for hosting intelligent virtual environments. We briefly introduce the notion of intelligent virtual environments and agents and present the advantages and drawbacks of their network variant. We demonstrate an architecture and topology for a network based approach and demonstrate its application on the Unreal Engine.

Key-Words: - Virtual Agents, Virtual Environments, Intelligent Agents, Distributed Agents, Network, Distributed Environments

1 Introduction

Intelligent Virtual Environments, (IVEs), are very demanding, in terms of computing power, due to the combination of realistic 3D graphics with artificial intelligence techniques. To overcome this, attempts have been made to migrate to a distributed IVE architecture : The systems DIVA, [1], and mVital, [6], are some examples of such approaches.

In this paper we suggest a different approach to such a distributed environment. Not unlike the DIVA and mVital systems, our architecture also focuses on the idea of decomposing the system into three different modules; one responsible for the manipulation of the agent, one for the "world" and one for the visualization of the environment. However, several changes have been introduced.

This paper is structured as follows: In Section 2 we present an introduction to intelligent virtual agents (IVAs), intelligent virtual environments and their network based approach. In Section 3, we present the architecture of the proposed system and its distribution over the network. In Section 4, we present a case study over the Unreal Engine. Finally, in Section 5, conclusions along with future work are discussed.

2 Network-Based IVA and IVE

2.1 Definitions

The term Intelligent Virtual Agent (IVA) is multifaceted and complex, requiring a definition depending on the context in which it's used. To expand the definition of intelligent agents given by Russel and Norvig, [3], we can state that:

An Intelligent Virtual Agent is a synthetic character that can perceive its virtual environment using sensors and act upon it by means of effectors, using AI techniques to form decisions. The term virtual refers to the graphical representation of the agent.

To further expand the definition, IVAs perceive dynamic conditions, act to affect them and use reasoning to interpret perceptions, solve problems, draw conclusions, and determine actions, [4].

IVAs display certain characteristics : an IVA can be autonomous, thus being able to operate without depending on humans, can interact with other agents, can interact with and interpret the environment and can dictate its own behaviour by defining goals to achieve.

The recent advancements in the realm of 3D graphics have led to the increasing use of Virtual Environments, [5] :

A virtual environment is an interactive 3D world, developed using computer technology, in which the objects have a sense of spatial presence.

The virtual world allows users to interact with its objects and incorporates physics, animations and sounds so as to render the whole experience realistic. Despite the high level of realism of today's virtual environments, the experience cannot be characterized as immersive, given the lack of interaction with other humans or computer-generated intelligent entities.

The meeting point of Intelligent Virtual Agents and Virtual Environments is an Intelligent Virtual Environment (IVE). It consists of a 3D world wherein IVAs can operate and interact with both humans and other IVAs. The increasing computing power contributes to the creation of visually realistic environments and allows for efficient application of advanced AI techniques. The agent can "sense" and

interpret the surrounding environment and act accordingly.

2.2 Network-based IVAs and IVEs

The advancement of networks and their widespread use, have laid the foundation for developing network-based IVAs and IVEs.

A Network-based IVA, while maintaining the basic IVA characteristics, also supports interaction with other remote IVAs, [2], and humans on shared, remote IVEs. Network based IVEs, are deployed in the form of a server hosting the virtual environment and allowing for either human or IVA clients to connect and operate in the virtual environment.

3   Architecture & Topology

Following the example of DIVA, [1], and mVital, [6], the system consists of the world server and one or more visualization and agent clients. The world server contains and manages the virtual environment, but is not responsible for visualizing it. It manages all the information regarding the world, its objects, object relations and its 'laws'. Furthermore, the world server is responsible for generating the agents' sensory input and for executing the agents' actions.

The visualization client displays the virtual world in real-time using 3D techniques.

The agent clients are responsible for interpreting the agents' senses, making decisions and taking actions.

The agents operate in 'Sense-Decide-Act' loops, while the world server operates in 'Receive-Update-Send' loops.

Specifications of the communication between the world server, the visualization client and the agent clients, as well as the format of the communication messages must be now given.

3.1   Communication Model

Amongst the modules of the proposed system, two types of communication solutions can be developed.

- Direct linking between the modules
- Network based approach

The first approach suggests the direct linking of the three modules of the system. Direct linking is the creation of a static library, or a dynamic link library, e.g. a dll for Windows, for each module, and combining them with a core application, thus producing the final system. The core application is responsible for the communication between the modules, distributing the messages through appropriate function calls. The modules exist in the application in the form of objects, the functions of which are executed via threads.

The second approach suggests the use of a network protocol as the means of communication between modules. Therefore, the modules have to be standalone, executable applications, possibly located on several different computers. The protocol used for the communication can be either the TCP/IP or the UDP protocol. Both require the opening of a socket, i.e. a listening and transmitting port, used for exchanging messages.

The simplest approach is the use of direct linking between the modules. Even though it is quite simple, easy to implement, thoroughly tested and used in numerous projects, it exhibits a significant disadvantage; the inability of scaling in a distributed environment results in the lack of distributed computing advantages, such as the use of multiple processors for handling the different tasks of the application.
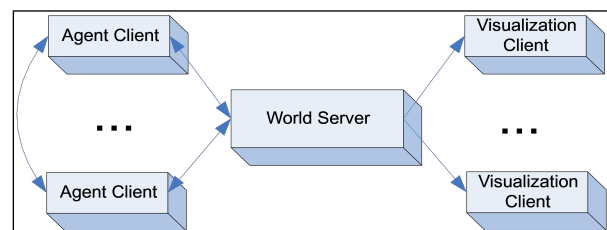

Fig.1 Topology of Network based communication among modules

On the other hand, the network based approach is natively capable of migrating to a distributed environment. The main advantages of the network-based implementation are that:

- the environment and the agents that live within are decoupled,
- the user can remotely access and interact with the IVE
- the distribution of computing power allows the implementation of more advanced graphics and AI techniques,
- the IVE can support multiple users simultaneously

Therefore, solutions to the above problems must be found and implemented. However, there are a numerous drawbacks to this approach, too. First of all, the messages passed over need to be formatted by the sender and analyzed by the receiver, via a predefined protocol. The above process is time consuming and requires system resources, such as network bandwidth and cpu-time. Moreover, the delivery time of the messages is not insignificant.

The delay between the dispatch and the arrival of a message can vary depending on network traffic, bandwidth and end to end distance. If the messages sent are relatively large and frequent the system will be delayed and the network overloaded.

In order to resolve the above problems we have introduced some techniques so as to overcome network limitations :

- Transmit only the required information, by distributing parts of the world server regarding the agent to the agent clients.

For example, if the agents' vision is implemented exclusively in the world server, the agent client would require querying the server continuously to receive visual stimulus. This results in frequent and large messages. By transferring part of the knowledge concerning the position of the objects to the agent client such queries are not required.
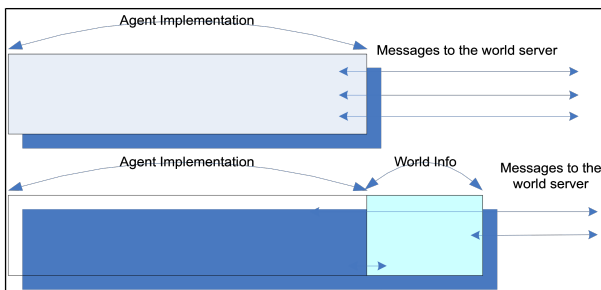

Fig.2 Agent client enhanced with information concerning the world

- Define more complex actions, and transmit them, instead of the elementary ones they are comprised of.

For example, the agents' motion consists of several small steps. Instead of transmitting these elementary actions to the visualization client and the world server, the agent can instruct the avatar to move to a specific position without caring about the intermediate steps.

- Connect each agent client, not only with the world server, but also with all the agent clients.

This results in reducing the workload of the world server, as well as the network traffic, because otherwise the world server would have to retransmit messages from one agent client to the others.

3.2 Sense – Decide – Act

Sensing is the ability of the agents to receive information about their environment. Sensing can be divided into the following categories:

- Un-requested events, which are sent to the

agents when their sensors are stimulated, i.e. an event notifying the agent about bumping on a wall.
- Continuous requests for sensory information, made by the agent, i.e. frequent queries concerning the objects inside the agent's field of view.

Sensing is performed in the world server. The first category of sensing produces messages which are short, and at the same time relatively scarce. Therefore, such messages do not create significant network traffic. On the other hand the second category of sensing produces messages which are rather large and frequent.

The fist category of sensing can be implemented in the world server. However, the second one must be implemented in the agent client. Therefore, the required bandwidth and cpu-time for the transmition is minimized. Moreover, as the necessary information, such as vision information, is located on the client, the querying is much faster, resulting in more accurate knowledge of the environment.

However, the agent client must be informed about changes in the environment, occurring from both other agents and the "laws of the world". The client can be notified by the world server for every change, or by another agent client which is responsible for the change. We followed the second approach, as it reduces the network's traffic. Direct connections between the agent clients are mandatory.

Deciding takes place in the agent client and is based on the interpretation of the sensory information received, the knowledge base of the agent and a BDI, Beliefs-Desires-Intentions, model. As stated above, for performance reasons, the client receives all the information concerning changes in the virtual world. However, for believability reasons, the agent is allowed to use only the part of the information that has already been sensed.

Acting is the "implementation" of the decisions of the agent. Actions can be broken down to two categories, those that affect the environment or other agents and those that affect the agent's state. These decisions can be instinctive, non-cognitive or cognitive. Messages concerning actions that belong to the first category are sent to the world server and the other agents, if required, and are visualized by the visualization clients. Actions of the second category are handled internally by the agent client.

3.3 Message Categories

The messages exchanged by the clients and the

server can belong to one of the following categories depending on the participating entities:

- Agent client to agent client messages.
- Agent client from/to world server messages.
- World server to visualization client messages.

The Agent client sends messages to the world server which contains commands about the intended action. This action will be executed in the virtual world. However there may be a delay between the reception of the message and the time that the message's instructions have been fully executed. Therefore, an additional categorization concerns:

- Latent messages
- Instant messages

For example, latent messages may be "move to" instructions, which are completed when the destination is reached. It is a possibility that the agent will not be able to reach the destination. The agent's client is notified about the event that blocked the agent, such as a bump on an obstacle, and that the instruction could not be executed. In the meantime, new instructions received, will either be executed in parallel, or override the execution of the previous instructions.

4 A Case Study over Unreal

The architecture described above has been followed to develop a network based, distributed agent system over the Unreal Engine. The Unreal Engine fulfills most of the requirements stated. However, in some cases, minor modifications had to be made.

As stated above, the system consists of three types of modules; the world server, the agent clients and the visualization clients. The Unreal Engine offers both approaches, i.e. direct linking and network based communication. The second approach was preferred, for reasons stated above.

The Unreal Engine provides both the world server and the visualization client in a single module. It is possible to connect the Unreal world server to additional Unreal visualization clients.

The Unreal Engine can handle up to five simultaneous active connections to agent clients. We have overcome this limitation by activating an agent client connection only when needed. The agent clients can be implemented in any programming language, which offers basic socket programming abilities. In our study, C# was used.

The sense-decide-act model, which was proposed above, is fully applicable to the Unreal Engine. During the initialization of an agent client, information regarding the objects of the world and

other agents are requested and transferred. Therefore, sensing can take place both in the engine's world server, as well as in the agent clients. In addition, the client establishes connections with all the other agent clients, so that they can communicate directly, without burdening the world server. The Engine also supports both latent and instant functions. They are used to implement corresponding latent and instant messaging.

5 Conclusions & Future Work

We have presented an approach for developing a network based system for hosting intelligent virtual environments.

We believe that the NAIVE approach addresses most of the problems found in distributed IVEs, providing an efficient framework, as shown in the case study over the Unreal Engine.

Moreover, NAIVE overcomes problems found in DIVA, [1], and mVital, [6], by following the basic architecture but extending it by the experience taken from the SimHuman system, [7].

We are currently trying to develop more lifelike agents, by taking advantage of the distributed nature and processing power of NAIVE. Moreover, we are trying to create scenarios, in which the agents exhibit competitive autonomous behavior, displaying their independence.

References:
[1] S.Vosinakis, G.Anastassakis, T.Panayiotopoulos, DIVA: Distributed Intelligent Virtual Agents, *Virtual Agents 99,* University of Salford, 1999, pp. 131-134
[2] Michael Wooldridge, *An Introduction to Multi-agent Systems*, John Wiley and Sons Ltd, 2002
[3] S. Russel, P. Norvig, *Artificial Intelligence: A Modern Approach,* Prentice Hall, 1995
[4] B. Hayes-Roth, An architecture for adaptive intelligent systems, *Artificial Intelligence*, Vol. 72, No. 1-2, 1995, pp. 329-365.
[5] S.Bryson, Virtual Reality: A Definition History, Lexicon Definition Supplement, NASA Ames VIEW lab, www.fourthwavegroup.com/fwg/lexicon/1725w1.htm
[6] G.Anastassakis, T.Panayiotopoulos, A System for Logic based Intelligent Virtual Agents, *International Journal On Artificial Intelligence Tools*, in press**,** 2004
[7] S.Vosinakis, T.Panayiotopoulos, A tool for constructing 3D environments with Virtual Agents, *Multimedia Tools and Applications Journal*, in press, 2004.