Adaptive Visualization over the Internet

K.N. Tsoi, and E. Gröller
{joe, meister}@cg.tuwien.ac.at

Abstract

This report gives an account on our work in adaptive visualization over the Internet ressearch. In this report we present our prototypes of adaptive visualization systems and an agent-based visualization model for network applications. We propose an intelligent agent system for adaptively dispatching of visualization processes between a cluster of computers based on the changing network transmission bandwidth and processing power. We also discuss the scientific contributions of our work to the current research in Interne based visualization system design.

1. INTRODUCTION

This technical report gives an account on our work in adaptive visualization via the Internet research and a proposed agent-based adaptive visualization model. We present an agent-oriented Adaptive Visualization Agents Model (AVAM) for Internet applications, which adapts itself to changing resource situations, such as computers processing power and network transmission bandwidth, and dynamically partition visualization processes and workload onto computer hosts (i.e. client workstation and visualization server) in a distributed heterogeneous Internet environment. In the subsequent sections of this report, an overview of the proposed project and its objectives, and its current progress will be discussed. In section 2, we will also give arguments for contributions and benefits that the proposed system may give to the field of “visualization via the Internet”. The related work of ours and other researchers in the fields of intelligent agents, and Internet-based visualization systems are discussed in section 3. Our proposed work for an adaptive visualization system model is elaborated in greater detail in section 4. Section 5 is the conclusion of this report and we also discuss the future work of our ongoing visualization agents project.

2. ADAPTIVE VISUALIZATION VIA THE INTERNET

In this section, we give the arguments for the contributions and benefits that our work may give to the field of “adaptive visualization via the Internet”.

2.1. Adaptive Visualization Agents Model for Internet Applications

Visualization systems based on contemporary client/server system architectures have limitations in terms of supporting interactive applications via the Internet as processes are statically mapped between client and server during the system design. Due to the dynamic nature of the heterogeneous Internet, it is rather difficult (impossible) for static distributions of visualization processes to balance the workload between client and server and to achieve optimal performance through effective exploitation of available resources at run-time. A more flexible software engineering approach is needed for developing adaptive Internet-based (including WWW) interactive 3D visualization systems. In this project, we adopted the agent paradigm. An agent-oriented Internet-based prototype system for displaying iso-surface extractions, using the Voyager^{tm} ORB 3.2 agent toolkit [see Appendix A for more information] is being developed. Visualization agents carry processes and can migrate themselves (code mobility) between a network of computers to accommodate to the bandwidth and processing power limitations based on the observed situation. The goal is to improve performance of visualization through minimizing network usage, reducing the communication overheads required and balancing the computational loads between client/server hosts.

2.2. The Current Research Challenges and Problems

Delivering quality interactive visualizations of 3D volume data via the Internet presents many challenges. We identified the following three main problematic topics that bring challenges to our research:

(1) Network transmission bandwidth and availability of processing power, are two key determining computational resources. Visualization processes, such as direct volume rendering and flow visualization, are usually computationally expensive. In addition, transmitting vast amounts of data (i.e. raw data and visualization results) requires sufficient network bandwidth in order to guarantee reasonable response time and quality interactive visualization service. The “frequent trips” style of communication in a client/server system requires additional bandwidth especially when security measures are enabled in the system. The conventional design usually sacrifices scene quality for better performance due to bandwidth and processing power constraints.

(2) The challenges of developing large scale visualization systems in a distributed Internet environment stem from the inherent complexity associated with distributed systems [3]. It is rather difficult to make design decisions during the system development for optimal partitioning of visualization processes and workload onto client/server machines, to minimize the communication latency, and to ultimately accommodate to the constraints of bandwidth and computational resource variations. This is due to the dynamic nature and heterogeneity of the Internet environment and of many uncertainty factors, such as configuration and computational capacity of a user workstation, and the availability of network transmission bandwidth at run-time.

(3) Limitations of the contemporary system architecture are one of the problematic issues. Most of the visualization systems follow a dataflow model of hierarchical centralized control to implement the visualization pipeline [13], and are based on the client/server model for network applications. Visualization processes are statically mapped and tightly coupled between client and server, and the interactions between various processes are usually too rigidly defined (the "hard-wired" engineering approach) at design time. Visualization processes are either mostly executed on the server and results are sent to the client workstation over the network, or users are allowed to download the software (either part of the codes or the whole package) from the server and execute them locally. The workload could easily become too high to support reasonable performance for timely response on a multi-user visualization server, while a user's workstation (client) may be idle most of the time although it has the capacity to share and perform some of the necessary computation. With a static distribution of the visualization processes, it is difficult (rather impossible) for the system to adapt to the changing resource situation and to balance the workload between user workstations (client) in order to maintain an optimal system performance because the system does not allow its codes to move between hosts at run time when the local resources are insufficient for the computation.

2.3. The Proposed Solution and its Scientific Contributions

To the problems at hand, we favor the agent paradigm to be a promising software engineering approach for adaptive distributed Internet applications over the conventional alternatives. The agent paradigm has many benefits for distributed Internet based visualization systems. It changes the nature of the work from data, processes, servers and clients (users). As in contrast to the contemporary hierarchical design of statically mapped and centralized controlled visualization processes, the agent-based approach provides a decentralized, loosely coupled solution for adaptive complex systems. There's no need for developers to design a centralized executive control for distribution of visualization pipeline processes over a network, and without regard to in advance the network topology, traffic patterns and system capacity of the user workstations. The benefits of adopting an agent-oriented system design are also due to the code mobility and autonomy. An intelligent agent is capable of making decisions independently (and locally) and automatically reacts to the changes in the environment based on the observed situations, which may not be foreseen at design time. Mobility allows processes (modules) to migrate between computer hosts at run-time and to communicate with other system resources (data and services) locally, which eliminates the need for making "frequent trips" style of communicate between client and server over the network, and hence greatly reduces the communication overhead. A local message is often faster than its remote equivalent. An agent can invoke a reusable visualization module (either remotely or locally) and migrate from machine to machine as part of its execution. The mobility allows visualization processes to be executed at the host where processing power is sufficient or data is located in order to reduce the network traffic, and hence increase the response time and increase throughput. Agent-oriented system design is an effective way to decompose complex distributed problems in terms of autonomous agents that engage

in automotive decision-making by itself through interaction with the computation environment and other software entities (agents). Every agent becomes a problem solving entity by itself. The decentralization of problem solving, in turn, reduces the system control complexity and communication overhead in a network environment at run-time, and hence, increases the response time of the system. The agent paradigm has been highly advocated as a promising next generation software model for complex and distributed systems [17].

We propose an *Adaptive Visualization Agent Model* (AVAM) for Internet applications. The goals are to endow individual visualization pipeline processes with independent abilities and minimal communication needs and provide globally coherent and efficient behavior. In section 4, we will elaborate more on AVAM.

2.3.1 Objectives

The objectives are to investigate and to develop

(1) An agent-oriented model for adaptive and interactive visualization via the Internet.

(2) Intelligent mobile visualization agents capable of balancing workloads between the involved computer hosts (i.e., client and server machines) through adaptively migration and execution of visualization tasks between hosts.

(3) Autonomous intelligent agents that monitor and extract information about the availability of operating system performance data, CPU/memory resource utilization thresholds, and network information (i.e., bandwidth and latency). These agents are also capable of communicating with other agents either synchronously or asynchronously.

(4) An optimal "cost-benefit" model for the constrained optimization problems [12, 14, 22]. This model will be used for developing a fuzzy-rule based system, which deals with the constraints of network transmission bandwidth and processing power.

(5) A NeuroFuzzy [20] hybrid decision-making system using fuzzy logic control [21] and neural networks. A reinforcement learning approach [36] using an unsupervised neural network approach will be adopted for training of the fuzzy control sets. This system is for judiciously and dynamically migration and execution of visualization tasks between computational hosts.

(6) A proof-of-concept working prototype agent-oriented visualization system for volume visualization applications, such as iso-surface extraction (e.g., marching cubes algorithm [27]) and direct volume rendering (e.g., ray casting [26]). The prototype system will also extend the AVAM architecture for supporting collaborative visualization.

(7) Evaluation study of our proposed agent-oriented visualization systems and systems based on contemporary design of statically mapped visualization processes. The outcome of this study will provide a set of quantitative performance data of both types of systems.

2.3.2 The Scientific Contributions

Considering the problems stated above (section 2.2), the central arguments for the benefits of our proposed solution and its scientific contributions are expressed and justified. As quantitative data that would show superiority of the agent-based approach over other contemporary techniques does not exist in literature so far, our arguments for the proposed solution that adopts the agent paradigm are qualitative in nature and deserve further investigation.

We believe that the outcomes of our research will advance the state of the art in the research of important areas of distributed interactive visualizations via the Internet, and will make the following distinct contributions.

(1) An Adaptive Visualization Agent Model (AVAM) provides a novel approach for developing high fidelity Internet-based interactive visualization systems, which are adaptive to the changing resources situations in a heterogeneous Internet environment without having to sacrifice the scene quality for better performance due to resource constraints. The proposed model is also extendable for supporting collaborative visualizations via the Internet.

(2) Our prototype system will demonstrate the capability of agent-oriented system design in supporting distributed and collaborative visualization via the Internet through dynamically dispatching and migrating visualization modules, and balancing the workload between hosts (client/server machines) based on observed resource situation at run-time of client/server.

(3) The working prototype systems will also contribute as a case study for researchers investigating various intelligent distributed visualization systems design techniques. Moreover, they will provide a basis of quantitative arguments for the benefits and feasibility of agent-oriented design for complex and distributed internet-based visualization systems.

(4) Currently, there are not studies reported in the literature about the application of NeuroFuzzy control systems for dynamically distribution of visualization processes among all involving computational hosts. Our prototype systems will contribute as a valuable case study material for researchers in the field.

(5) The insights and experiences gained in this work and the evaluation results will provide quantitative data for the researchers and software engineers in the field facing similar problems to assess whether an agent-oriented approach is feasible for them.

3. RELATED WORK

Below, in the context of our proposed study, we give a brief overview of the related work in Web-based visualization systems, Collaborative visualization systems, and intelligent agent systems for Internet applications.

3.1. Web-based systems (based on a client/server architecture)

Research efforts in Internet-based visualization have addressed various system design issues and investigated the flexibility of delivering interactive 3D visualization via the Internet. However, the majority of previous work assumes that the distribution of visualization processes is a design-time problem. And they do not specifically address the need for a flexible system architecture that effectively accommodates to the constraints of network bandwidth and processing power. These systems typically have visualization processes (pipeline steps) statically mapped either all on the client side or on the server side (while the client displays the resulting 3D scenes). At the beginning of either session (via a Web interface), users of these systems will either download/execute only part of the visualization codes (e.g., the display module) or entire code on to the local workstation.

These prototype systems are usually Web-based Java applets or use VRML [2, 24, 30, 32, 35, 37]. Users of [34] can access a set of commonly accessible visualization tools and techniques through an intuitive Web interface. Web-VIZARD [10, 11] is a Web-based GIS (Geographic Information System) for geographic data visualization using JAVA technology, which provides a set of APIs for integration of third party GIS systems as an Intelligent GUI. The aim of the system is to provide intelligent GUI for making system functionality more accessible to the end users. The NOVICE system [18] mainly emphasizes on the networked technologies, and provides a set of extensible web-based visualization tools for medical visualization within a high performance-computing environment. VizWiz [30] provides simple but innovative visualization services that allows users to visualize different data types using multiple visualization techniques. InVis [28, 29], focuses on the interactivity and adaptive (server) processing power utilization issues by incorporating a real-time control optimization mechanism for quality interactive visualization using an interactively adapting progressive refinement technique. It also provides a set of tools for parallel processing for multiple data types, such as volume data, geometric objects and hybrid data. Some systems adopted a “Thin” or “Fat” Client service mode [19] to balance the workload between the user’s workstation and the visualization server. “Thin” client means most of the visualization-steps are processed at the visualization server, while “Fat” client has most of the visualization-steps processed at a client workstation. More recently, a study [6] about Web-based iso-surface extraction techniques demonstrates a number of static distributions of visualization modules for optimal processing power utilization between client and server.

3.2. Agents and Applications in Visualization

An agent is a computational entity, which acts on behalf of other entities in an autonomous fashion. An agent performs the actions with some level of pro-activity

and reactiveness. An agent also exhibits some level of key attributes of learning, cooperation and mobility [16]. Some systems in real-time reasoning and process control are using a cooperative agent design [15, 25, 31]. There are so far a few visualizations systems that are based of the agent-oriented design. Researchers of the SurfaceMapper project [9] developed a multi-agent system for 3D scientific volume data interpretation. SurfaceMapper is a community of cooperative agents, which automatically locate and display interpretations of 3D scientific data from a store of vast volumetric data. These cooperative agents are namely *Segment Agents*, *Curve Agents* and *Surface Agents*.

3.3. Collaborative Visualization In a Network Environment

This section gives a short overview of the collaborative visualization systems reported in the literature. However, these studies are mostly based on the client/server design and do not explicitly address the design issues of adapting to the changing computational resources at run-time.

The main goal of TeleInViVo [4] is to facilitate therapy planning and treatment, medical training, surgery, and diagnosis using real-time visualization in a distributed environment. The physicians can exchange and manipulate data sets via ISDN or ATM networks. It supports collaborative visualization and exploration of volumetric data including computed tomography, magnetic resonance imaging and PET scans. TeleMed [33] is a platform independent system developed in Java and using the CORBA architecture. TeleMed is basically a multimedia patient records management system, which dynamically unites graphical patient records. It allows multiple user accesses to the database and performs tasks such as radiology examinations in real-time. CORBA can be used for developing agent-oriented systems, but this feature is not implemented in the current version of TeleMed. Shastra [1] is another Internet-based multimedia system, which supports collaborative and distributive visualization through implementation of two distributed visualization algorithms. The algorithms consist of a collection of inter-operating tools, which support managing, communication and rendering facilities. Users of Shastra can use a rendering and visualization tool called Poly for graphical objects manipulation, rendering and visualization. SDSC_NetV [5] is an experimental system. It is developed as a distributed system with advanced rendering techniques and exhibits stereo images. The design goal of this system is to overcome the performance problems with processing large volume data in a shared environment. It has a mechanism for managing the available resources for volume rendering in a network and allowing access by users either remotely or locally.

3.4. Adaptive Dynamical System Visualization

This is a work done preliminary at the beginning of our adaptive visualization agents research. The work is based on two phases, which aim to develop a prototype system for 3D visualization of numerical simulation of the Lorenz equations and to adaptively adjust the output of polygons produced by the visualization based on the changes in processing power (CPU and memory availability). The prototype system

for visualization the Lorenz equations is developed using the Java 3D toolkit. Figure 1 depicts the visualization output of the Lorenz equations.

Below the Lorenz system is given as first order differential equations:

$$dx/dt = sigma\ (y-x)$$
$$dy/dt = rho\ x - y - xz$$
$$dz/dt = xy - beta\ z$$

These equations are integrated using a fourth order Runge Kutta method for the parameter values: *sigma* = 10.0, *rho* = 28.0, *beta* = 2.6667.

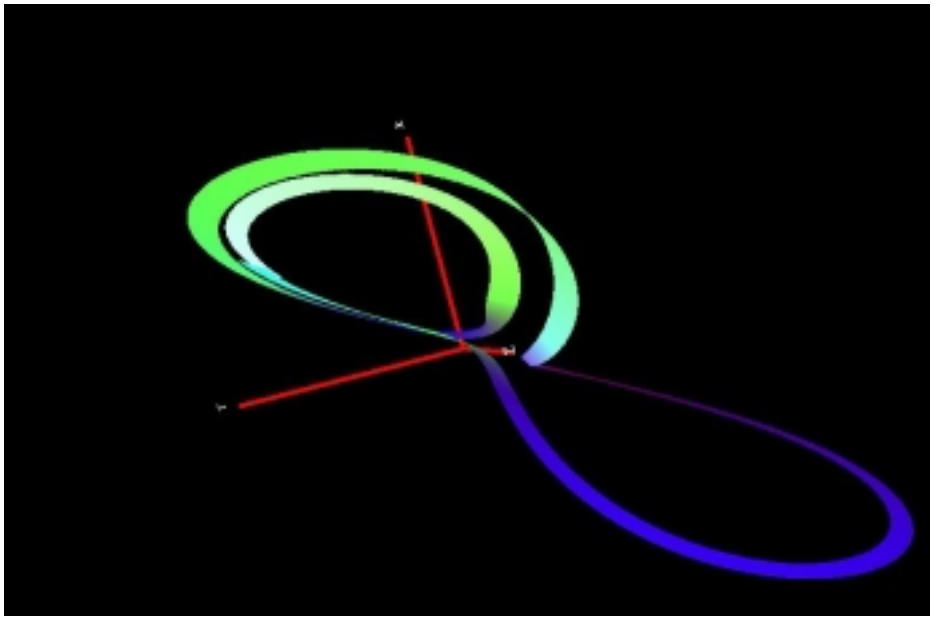

Figure 1. 3D visualization of the Lorenz equations

Based on the prototype visualization system for the Lorenz equations, we dynamically adjust the number of polygon outputs. The prototype system records the time (in ms) required for producing a certain number of polygons. At the beginning of a visualization session, the program first profiles how many polygons can the system produce in 1ms with the current system resource situation (CPU level and memory availability). It then determines how many polygons to produce next by multiplying the user's desired response time (in ms) with the number of polygons the system can produce. The equation below explains the working of the system.

$$NP = (TP/T)*UT$$

NP denotes the number of polygons for the next output, while *TP* is the total number of polygons of the previous output. *T* denotes the time used for the previous output and *UT* denotes user's desired response time. Figure 2 is a screen capture of the output of the system.

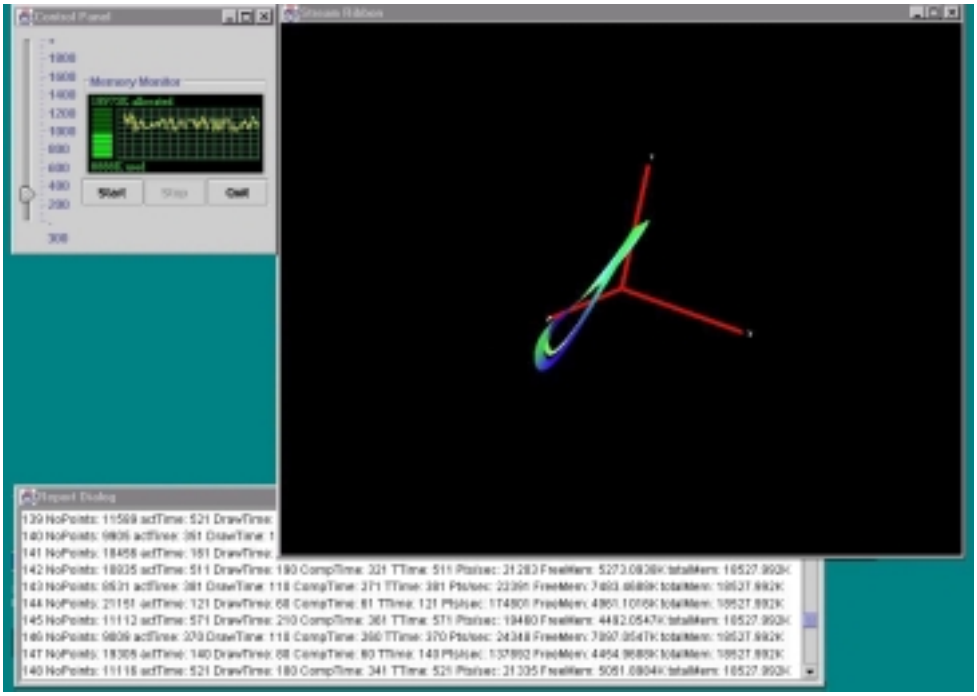

Figure 2. Screen capture of the dynamically generated polygon output.

4. THE PROPOSED AVAM

This section describes our proposed Adaptive Visualization Agents Model (AVAM). We have investigated an adaptive visualization agent's model for intelligent distributed visualization applications in the Internet environment. A prototype system using agent-oriented design is proposed and is being developed.

4.1. Motivations

Judicious adaptation of visualization processes to the available transmission bandwidth and processing power can ameliorate the impacts of many uncertainties and the underlying constraints of networked computation resources. Motivated by the non-uniform quality of service in heterogeneous Internet environment and non-uniform computing ability across computer platforms, we focus our research in devising a visualization system that can dynamically adapt and accommodate to the changing resource situations and constraints. We envision and wish to design an adaptive Internet based visualization agents system that is robust enough to guarantee quality interactive visualization services without sacrificing scene quality for increased responsiveness in a heterogeneous Internet environment. To turn our vision into reality, we need a careful designed system model and architecture.

4.2. An Overview of the Proposed System

The proposed *Adaptive Visualization Agent Model* (AVAM) for Internet applications provides an abstracted conceptual framework for designing agent-oriented visualization systems, which interact with the environment (and agents) and makes decisions at run-time locally based on the observed situations to fit to the prevailing circumstances of the system. An implementation of such agent system allows dynamical migration of visualization pipeline entities between client/server machines based on the server load, network latency, processor speed and computation power of the local machine, user's requirement for the quality and interactive performance.

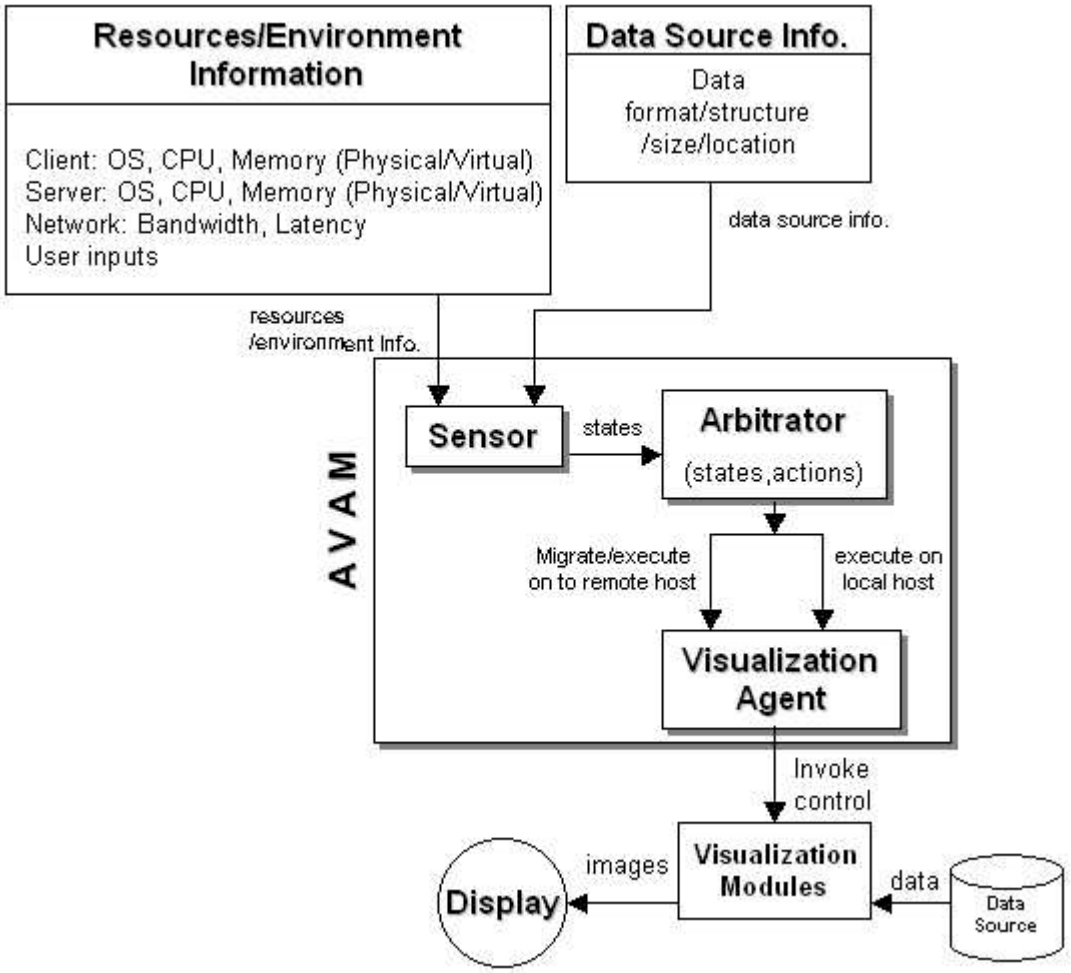

Figure 3. A Logical Overview of AVAM

Figure 3 depicts AVAM. This model takes the resources/environment information and data source information as inputs and actions/controls as outputs. We decompose AVAM into three main sub-systems based on their functionalities:

(1) *Sensor* monitors computing resource and network transmission bandwidth availability, receives user inputs for visualization parameters and file

information about the volume data set (such as file format, structure, size and location).

(2) *Arbitrator* is a decision-maker for a visualization agent. Based on the information collected by *Sensor*, it determines whether an agent should execute a visualization step locally or migrate it on to a remote host for execution.
(3) A *Visualization Agent* represents a designated mobile/autonomous visualization pipeline entity. It invokes visualization modules and carries out the decisions made by the *Arbitrator*.

The *Sensor* extracts and generalizes the information (states) regarding the computing resources and network bandwidth information, the targeted data source for visualization (e.g., data format, structure, size and location), and receives user inputs of visualization parameters. The *Arbitrator* makes decision about on which host the visualization agent should execute its visualization pipeline step.

4.3. AVAM Visualization Agents Architecture

Figure 4 depicts a visualization agents architecture based on AVAM. This architecture illustrates an example of a multi-agent environment involving main hosts (visualization server) and user hosts (client machines). The main hosts consist of Host A (the main visualization application host) and two computational slaves (Host B and C) located within the same local area network. User host A and B receive visualization services from the visualization server. The main visualization host is the home of visualization agents, which maintains a library of the visualization modules. Each visualization agent has an embedded *Arbitrator*. A stationary *Network Monitor Agent* resides on the main host (Host A), which periodically monitors the bandwidth situation of the connection between each of the user client machines. Each of the computation hosts involved has a stationary *System Resource Monitor Agent*. The dashed lines represent the migration of visualization agents between the hosts within the architecture. Scenarios of mobile autonomous visualization agents are illustrated in section 4.4. The key components of this architecture are described as follows:

Network Monitor Agent and ***System Resource Agent*** (a.k.a. sensor agents) are information collection and extraction systems. The sensor agents may be stationary autonomous agents. A stationary agent is non-migratory by design as part of its execution. The *Network Monitor Agent* resides on the visualization server (main host) and monitors the network bandwidth situation between the main host and client hosts, while the *System Resource Agent* resides on every host machine and monitors the local availability of processing power, such as CPU/Memory. They report these collected information to visualization agents upon request at run-time.

Arbitrator is an embedded decision-making unit within a visualization agent. Based on information about the changes in the computational environment reported by the Sensor Agents, the Arbitrator decides for a Visualization Agent which visualization modules to invoke and where (host) the agent should migrate to for execution of the visualization processes. Arbitrator maintains a vector of “state/action” fuzzy rule sets

for appropriate actions performed according to an agent's prevailing circumstances at run-time.

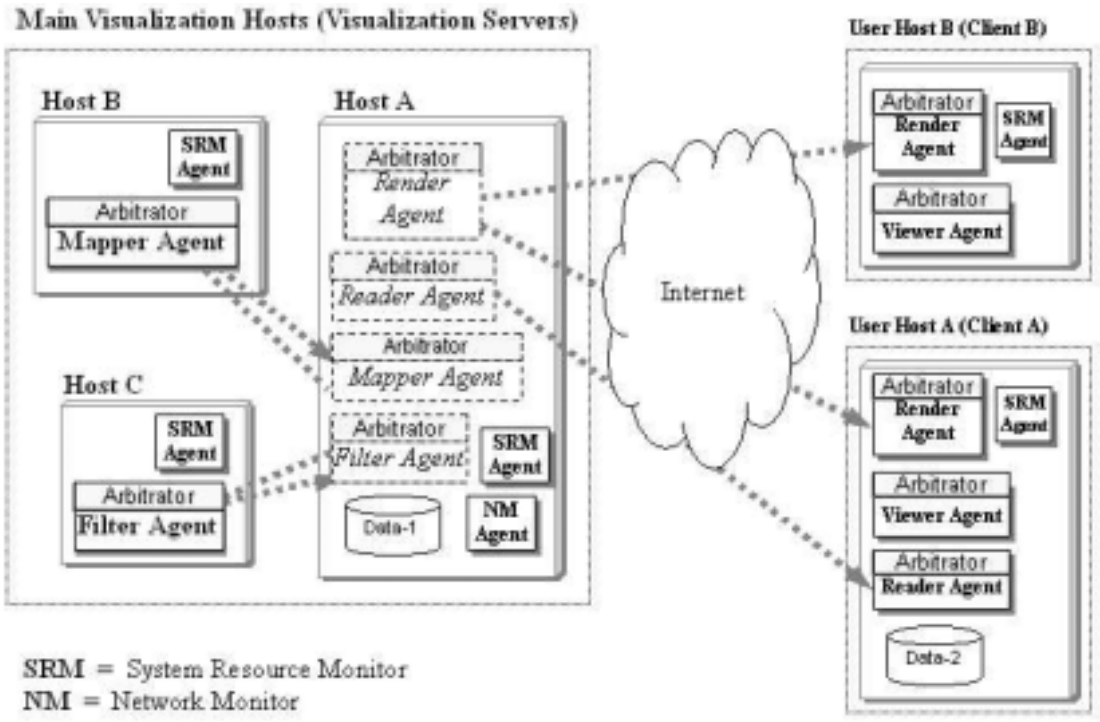

Figure 4. Visualization Agents Architecture

Visualization Agents are a collection of autonomous agents (except for Viewer Agent, which is designated to be stationary). A visualization agent consists of an embedded arbitrator and is capable of making localized decisions, functions for invoking and detaching visualization modules, and functions to handle code mobility (migration to a remote host) and execution of visualization modules. A visualization agent can invoke and detach modules either locally or remotely over a network at run-time. These visualization agents are modeled based on the conventional visualization pipeline entities, namely *Reader Agent*, *Filter Agent*, *Mapper Agent*, *Render Agent* and *Viewer Agent*. A visualization agent moves the visualization modules to a remote host or where the data is located (i.e., *Reader Agent* moves to user host A to access the data) for execution if the local computation resource is insufficient or the network latency is too high. Among the visualization agents, *Viewer Agent* acts as an interface agent between the visualization system and the users, for user's visualization requests and parameter input (i.e., change of iso-value and view point). While the rest of the visualization agents are mobile in nature, the *Viewer Agent* is not mobile (but may be downloaded to a user's workstation at the beginning of the visualization service).

4.4. Mobile Visualization Agents Scenarios

Figure 5 depicts four possible scenarios of visualization agents migrating between host machines (client/server). We used a marching cube algorithm application (iso-

surface extraction from a volume dataset) as an example in these scenarios. In these scenarios, we assume that all client hosts involved have graphics adapter for rendering 3D polygonal models so that *Render Agent* can move to the client host at run-time to produce rendered images.

In scenarios (a) and (b), data is located at the client host and the network is too slow for sending volume dataset of large size to maintain fast interaction (due to available transmission bandwidth is insufficient or latency is too high) at run-time. *Reader Agent* migrates to the client host, determines the format of the volume data and loads the data using a data-reading module of a specific file format. *Filter Agent* takes the raw data read by *Reader Agent* to filter out cells from the volume dataset that are intersected by iso-surface. *Filter Agent* then interpolates along cell edges and carries the results back to the main visualization host over the network. *Mapper Agent* at the main visualization host then computes the iso-surfaces (iso-surface extraction) and produces triangle/polygon (geometry transformation) using the interpolation values and marching cube cases passed from the *Filter Agent*. It then invokes a projection module to project the triangles/polygons data to the viewing plane and maps them to the screen coordinates. *Mapper Agent* also invokes a rasterization module to render screen coordinates of triangle vertices along with the colors at the vertices into frame buffer. The *Viewer Agent* receives rendered images from *Render Agent* and displays them to the users. *Render Agent* can also carry the triangles as its current computation state to the client and does the rendering locally if the client host has sufficient resources and the network latency is high. This guarantees immediate display of the rendered images and avoid transferring them over the congested network.

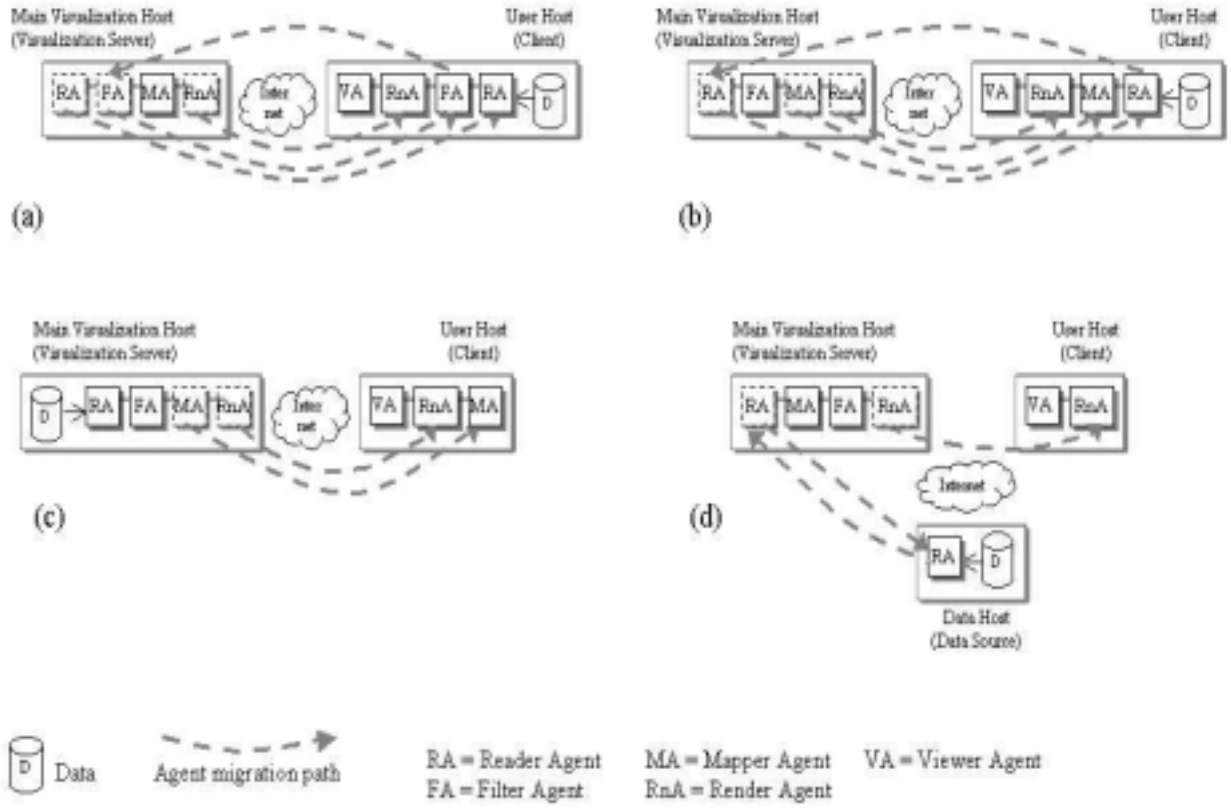

Figure 5. Visualization Agents Migration Scenarios

In scenario (c), the volume dataset is located on the main visualization host. Assuming the client host has the capacity to do the mapping and rendering tasks. *Render Agent* and *Mapper Agent* both migrate to the client host. Triangle setup and rendering are both done on the client. In this scenario, local interaction for viewpoint changes without delay due to network latency is allowed. A large volume dataset stored on the remote server can be visualized over the latency network without the need of transferring the whole set of data to the client machine.

In scenario (d), the volume dataset is stored on a remote data host. *Reader Agent* moves to the data host, and moves back to the main visualization host with the retrieved data. This scenario demonstrates the flexibility of mobile agents in the situation where the location of the data cannot be determined at the design time.

4.5. Agents Communication

In this section, we propose a "mailbox" approach as communication mechanism for sensor agents and visualization agents. To choose an efficient communication method for a multi-agent system is a challenge. We therefore investigated "Procedure call", "Callback" and "Mailbox" mechanisms. With the callback mechanism, agent A calls agent B and then continues on with its tasks. When agent B has done whatever it was asked to do, it calls back agent A and passes the result. Agent A has to stop the current job to deal with the callback (windows system, Java AWT and Swing are all using callback). Procedure call works in the way as agent A calls agent B and waits until B has some result. Agent B may work for a while and realizes that it needs something else from Agent C and then it calls C and waits for C to reply. Unlike the callback mechanism, procedure call works in a sequential mode; it is easy to be followed. With the mailbox mechanism, agent A asks agent B and tells B to put the finished result in its mailbox. A then goes about its business checking the mailbox periodically to see if B is finished. Figure 6 depicts the communications of visualization modules in a pipeline.

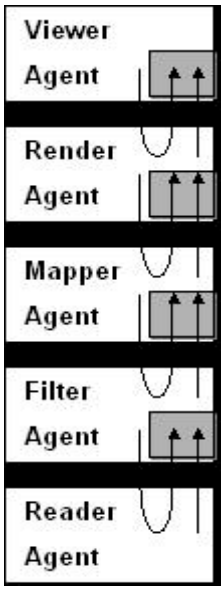

Figure 6. Mailbox Mechanism

We favor the mailbox method over the other two communication mechanisms because of two compelling reasons.

(1) It allows asynchronous processing: Agents are, by their very nature, autonomous, independent, distributed software entities. In an Internet environment, with its inherent delay (network latency), agents cannot afford to wait for something they need at some future point (compared to the procedure call mechanism).

(2) Easy to trace flow of execution: the mailbox method avoids hiding flow of execution. In a distributed system with processing often spread across a number of machines in a network, debugging and tracing of the flow of execution is difficult (as with the callback mechanism).

Although the mailbox mechanism is pragmatically more difficult to implement than either of callback and procedure call mechanisms, it allows for asynchronous processing while at the same time avoiding the problem of a confusing flow of execution. These two are important factors in distributed systems.

4.6. Implementation Environment

The test environmetn consists of a cluster of high-end graphics workstations (to be sufficient to provide a network of host PCs which provide various processing powers of different work loads), as in addition to the existing visualization servers for an implementation and testing enviroment of our prototype systems within the BandViz project. Our visualization agents migrate among the network for execution of computation according to the changing network bandwidth and system resource situations on the host computers. Network traffic patterns similar to or close to the real life Interent environment is being simulated through intentionally sending of vast amount of video files over the network between host PCs within the LAN of our test environment (so that it will not interfer with the regular LAN).

5. CONCLUSION AND FUTURE WORK

Voyager provides an agent framework for mobility and autonomy of code. We have so far explored the partial functionality of this toolkit. There are still technical difficulties of moving our visualization codes around the network with Voyager due to version incompatibility between Voyager and JDK.

The working prototype of Java mobile visualization agents should have the following characteristics:

- Mobility: Agents can carry their codes, data and execution state with them from one computer to another across the network.
- Autonomy: Algorithms implemented in the code of agents enable them to make local decisions on what to do, where to go and when to go.
- Concurrency: Multiple agents can be dispatched simultaneously to accomplish various parts of a task in parallel. For example, processing of multiple slices of a volume data set.
- Local interaction: Mobile visualization agents interact with local entities, such as data source and stationary agents (sensor agents), through method invocation, while interaction with remote entities is by message passing.
- Rapid response: An agent can visit several hosts, interacting with local entities at each host, and can return to its home base in only a few seconds.

As our work on AVAM has not yet produced any quantative results for comparision against conventional visualization system design for network applications, we can only argue th benefits of AVAM and its contributions to the field of Internet-based visualiation on a qualitative basis. We are confident that agent design will take an important role to advance the state of the art for visualization via the Internet. In the near future, we plan to continue implementing AVAM and realize our vision of an agent-based visualization system for networked applications.

6. REFERENCES

1. Anupam, V. et.al.: *Distributed and Collaborative Visualization*. Computer, Los Alamitos, CA, Vol.27(7). pp.37-43, July 1994.
   at http://www.ticam.utexas.edu/CCV/projects/shastra/
2. Brodlie K., et al, *Web-based Visualization: A Client Side Approach*. IEEE Workshop on Distributed Visualization Systems, 1998.
3. Brooks, F.P. *The Mythical Man-Mouth*, Addison-Wesley, Reading, Mass., 1975.
4. Coleman, J.: *TeleInViVo*, Fraunhofer – Center for Research in Computer Graphics.
   http://www.crcg.edu/projecs/medvis/TeleInViVo/Publicity/cyberedge.html
5. Elvins, T.T.: Volume Visualization in a Collaborative Computing Environment, Computer&Graphics, Oxford, Vol.20(2), pp.219-222, 1996.
6. Engel K, Westermann R, and Ertl T, *Isosurface Extraction Techniques for Web-based volume Visualization*, Proceedings IEEE Visualization 1999, IEEE Visualization Conference 1999.
7. Forslund, D.: *TeleMed*, http://www.acl.lanl.gov/TeleMed
8. Forslund, D.: *CORBAmed*, http://www.omg.org/homepages/corba med/
9. Gallimore, R.J., et al, (1998) *3D Scientific Data Interpretation using Cooperating Agents*, Proc. 3rd Int. Conference on the Pratical Applications of Agents and multi-Agent Systems (PAAM-98), London, UK, 47-65.
10. Goebel, S., *WebVizard: Intelligent System for Geodata Visualization and CBT in WWW*, Proceedings of Computer Graphics International'98, Los Alamitos, CA: IEEE Computer Society Press, 1998, pp.113-122.
11. Goebel, S., *WebVizard: Intelligent Geodata Visualization with JAVA*, http://www.igd.fhg.de/igd-a5/projects/web_viz.html.en, IGD, Fraunhofer, Germany, 1999.
12. Gobbetti E, and Bouvier E, *Time-Critical multiresolution Scene Rendering*, Proceedings IEEE Visualization 1999, IEEE Visualization Conference 1999.
13. Haber, R. B.; Mcnabb, D. A., *Visualization Idioms: A Conceptual model for Scientific Visualization Systems*, Nielson, G.M.; Shriver, B.D. (Eds.), Visualization in Scientific Computing, IEEE, pp.74-93, 1990.
14. Hoppe H.. *Efficient implementation of progressive meshes. Computers & Graphics*, Vol. 22, No. 1, 1998, pp. 27-36. (Other version available as MSR-TR-98-02.)
15. Ingrand, F.F.; Georgeff, M.P., and Rao, A.S., (1992) *An Architecture for Real Time Reasoning and System Control*, IEEE Expert 7 (6).
16. Jennings, N.R.: *On Agent-based Software Engineering*, Artificial Intelligence, Vol.117(2000), pp.277-296, Elservier Science, UK., 2000.
17. Jennings, N.R. and Wooldridge M.; *Agent-oriented Software Engineering*, in: J. Bradshaw (Ed.), Handbook of Agent Technology, AAAI/MIT Press, 2000.
18. Jern M., *NOVICE: Network-Oriented Visualization in the Clinical Environment*, A Work in Progress Project, http://www.man.ac.uk/MVC/projects/NOVICE/, MVC, 1999.
19. Jern M., *"THIN" vs. "FAT" Visualization Client*, Computer Graphics International, IEEE, pp.772-788, June 1998.
20. Khan, E. and Unal F.A.: *A Fuzzy Finite State Machine Implementation Based on a Neural Fuzzy System*, IEEE 1994, pp.1749-1754.

21. Klir, G.J. and Yuan B.*: Fuzzy Sets and Fuzzy Logic: Theory and Application*, New York. Prentice Hall, 1995.
22. Klosowski J. T., and Silva C. T., *Rendering on a Budget: A Framework for Time-Critical Rendering*, Proceedings Proceedings IEEE Visualization 1999, IEEE Visualization Conference 1999.
23. Lefer, W., *A Distributed Architecture for a Web-based Visualization Service*, Proceedings of 9th Eurographics Workshop on Visualization in Scientific Computing, Blaubeuren, Germany, Apr. 1998.
24. Lefer, W., and Pierson J-M, *Dynamic Distributed Environment for Data Visualization on the World Wide Web*, Proceedings of IEEE Workshop on Distributed Visualization System (in conjunction with the IEEE Visualization'98 Conference), Research Triangle, NC, Oct. 1998.
25. Lesser, V.R. and Corkill, D.D., (1981) *Functionally Accurate, Co-operative Distributed Systems*, IEEE Transactions on Systems Man and cybernetics 11 (1) 81-96.
26. Levoy, M.: *Efficient Ray-Tracing of Volume Data*. ACM transactions on Graphics, New York, Vol.9, No. 3, July 1990.
27. Lorensen, W.E.: Cline, H.E.: Marching Cubes: A High resolution 3D Surface Construction Algorithm, Computer Graphics, New York, Vol.21, No.4, July 1987.
28. Meyer, J., et al., *InVis: Interactive Visualization of Medical Data Sets,* at the http://davinci.informatik.uni-kl.de/~jmeyer/InVIS/eng/invis.htm, Department of Computer Science, University of Kaiserslautern, Germany, June 1997.
29. Meyer, J., et al., *Interactive Visualization of Hybrid Medical Data Sets,* Proceedings of WSCG'97 The Fifth International Conference in Central Europe on Computer Graphics and Visualization'97, 1997.
30. Michaels C.; *VizWiz: A Java Applet for Interactive 3D Scientific Visualization on the Web,* http://www.sdsc.edu/vizwiz/, Proc. IEEE Visualization'97, pp.261-267, 1997.
31. Munir-ul M. Chowdhury and Yun Li, *Evolutionary Reinforcement Learning for Neurofuzzy Control*, Technical Report, CSC-96020, Faculty of Engineering, Glasgow G12 8QQ, Scotland, UK, 1997, URL: http://www.mech.gla.ac.uk/Research/Control/Publications/Reports/csc96020.ps
32. Science3D, *Science3D.com: Learn about Science over the Web with 3D Animations and Visualization*, http://www.science3d.com/, April 1999.
33. TeleMed, User Manual: *Graphical patient record (GPR) Window*, at http://www.acl.lanl.gov/TeleMed
34. Trapp J. and Pagendarm H-G., *A Prototype for a WWW-based Visualization Services.* 8th EG Workshop on ViSC, Boulogne sur Mer, 28-30 April, 1997.
35. Ulmer, H.; and Lindenbeck, Ch.: *Geology Meets Virtual Reality: VRML Visualization Server Applications*, Proceedings of the WSCG'98 6th International Conference in Central Europe on Computer Graphics and Visualization'98, Plzen, Czech Republic, 1998.
36. Watkins, C.: *Learning from Delayed Rewards*, Thesis, University of Cambridge, 1989, England
37. Wood, J.D.; Brodlie, K.W. and Wright H., *Visualization over the World Wide Web and its Application to Environmental Data*, Proceedings IEEE Visualization'96, pp.81-86, 1996.

38. Wooldridge M.: *Agent-based software engineering*, IEE Proc. Software Engineering 144 (1) (1997) 26—37

Appendix A. Voyager Agent Toolkit

ObjectSpace Voyager™ ORB 3.2 (Object Request Broker) is a Java agent-enhanced object request broker (ORB) that provides a distributed system architecture for developing mobile and autonomous software. The toolkit can be found at the company web site of ObjectSpace at **http://www.objectspace.com**