

# A REAL-TIME CORBA MIDDLEWARE FOR NETWORKED SURGICAL SIMULATIONS \*

Vincenzo Liberatore, M. Cenk Çavuşoğlu, Qingbo Cai  
Division of Computer Science  
Case Western Reserve University  
10900 Euclid Avenue, Cleveland, Ohio 44106  
vincenzo.liberatore@case.edu, cavusoglu@case.edu, qingbo.cai@case.edu

## Abstract

Virtual environments are a promising new medium among the simulation methods for surgical education. The accessibility of surgical virtual environments can be substantially extended by network communications. However, the network settings pose some non-trivial problems in terms of bandwidth limits, delays, *etc.* for distributed surgical virtual simulations.

In our previous work [3], we presented *GiPSi*, an open source/open architecture framework for developing surgical simulations. In our ongoing development [5], we extend GiPSi to network environments. This network extension involves the development of a middleware module (*GiPSiNet*) to remediate for the lack of network *QoS* and to enhance the user-perceived quality of a networked simulation. In this paper, we introduce real-time CORBA components to the design of the GiPSiNet middleware and describe the techniques to provide timely data delivery over the network.

## 1 Introduction

Among the various simulation methods for surgical education, virtual environments are a promising new medium. With virtual environments, a user can perform surgery on a simulated patient by manipulating simulated organs using simulated surgical instruments (i.e., haptic devices). An example of a surgical simulator is given in Figure 1, where a simulator senses a user's input (the position, orientation, etc. of a haptic device) and respond with haptic force feedback to the user in real-time.

The accessibility of surgical virtual environments can be substantially extended by network communications. The resulting network extension will enable continuing education and advanced training over wide geographical areas. However, the network settings pose some non-trivial problems for distributed surgical virtual environments because: (i) the physical communication platform introduces constraints in terms of delays and bandwidth limits; and (ii) haptic devices should be updated at a rate of 1k Hz or higher to produce a stable servo loop, but haptic data traffic cannot expect any special handling within a best-effort network such as the Internet and consequently are subject to delays or packet losses due to congestion. Therefore, in an end-to-end sense-and-respond real-time system such as the distributed surgical virtual environments, the network issues can greatly impair the user-perceived simulation performance (fidelity and realism). Hence, the

---

\*URL: <http://vorlon.cwru.edu/~vx111/>.

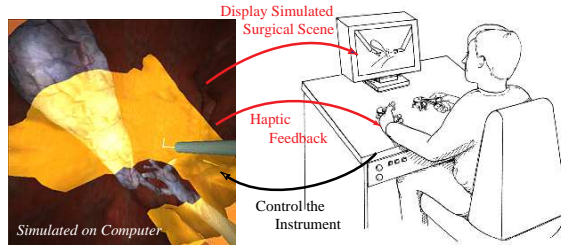


Figure 1: Surgical simulator concept. Simulation snapshot courtesy of F. Tendick [7].

methods to implement an effective remote interaction are especially critical to the quality of a networked surgical simulation.

In our previous work [3], we presented *GiPSi* (General Interactive Physical Simulation Interface), an open source/open architecture framework for developing surgical simulations. GiPSi works on individual workstations and, in our ongoing development [5], we have designed a middleware module *GiPSiNet* to extend GiPSi to network environments. GiPSiNet acts as an intermediary between applications and the network and takes actions to remediate for the lack of network *QoS* (Quality of Service) [6] and to enhance the user-perceived quality of a networked simulation.

In the GiPSiNet middleware, we adopt a real-time CORBA (Common Object Request Broker Architecture) implementation, i.e., TAO (The ACE ORB), for network communications. Wrappers around the TAO classes are also provided to encapsulate the network functionalities. Moreover, we will further revise TAO to make it more suitable for real-time systems by, for example, implementing a UDP-based RTP/RTCP (Real-time Transport Protocol/Real-time Transport Control Protocol) as a pluggable inter-ORB protocol.

**Organization.** In Section 2, we briefly introduce the features of CORBA and TAO. Section 3 describes how the GiPSiNet middleware uses TAO to simplify the remote interaction implementation, and how GiPSiNet revises TAO to improve its suitability for real-time systems. Section 4 concludes this paper.

## 2 CORBA and TAO

The CORBA specification is written and maintained by the OMG (Object Management Group) and provides flexible abstractions and concrete services for practical solutions to the problems associated with distributed heterogeneous computing [4]. CORBA automates many remote invocation tasks such as object registration and activation, concurrent request control, parameter marshalling and demarshalling, request dispatching, etc.

However, obstacles exist to apply CORBA to real-time systems, since many real-time requirements are associated with end-to-end system-wide aspects that transcend the layering boundaries traditionally associated with CORBA [1]. To provide end-to-end predictability for CORBA operations, OMG standardizes a Real-time CORBA specification, where standard features such as real-time ORB and priority mappings are defined. Among the real-time CORBA implementations, TAO is freely available and is widely used in many military and industrial systems such

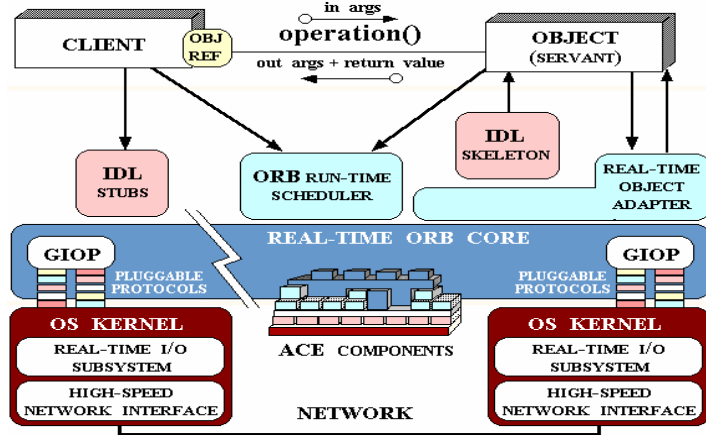


Figure 2: The TAO software architecture [1].

as the manned and unmanned real-time avionics mission computing embedded systems at Boeing/McDonnell Douglas. The ORB core of TAO is based on the ACE (Adaptive Communication Environment) framework and the TAO software architecture is shown in Figure 2.

### 3 GiPSiNet Design

The GiPSi framework currently provides an API between the simulation kernel and I/O (haptic device, visualization). To extend the kernel-I/O API to network environments, the GiPSiNet middleware is added between the kernel and the I/O module to encapsulate all networking functionality. The resulting simulator involves two communication end-points: (1) the haptics and visualization *client*, which interacts with the end-user through haptic devices and visualization interfaces, and (2) the *server*, where the simulation kernel runs and the physical models are numerically simulated.

An objective of GiPSiNet is to isolate the GiPSi application developers from the underlying network implementation details (e.g., message passing, remote procedure calls, etc.). To achieve this goal, in GiPSiNet, we define the common objects such as remote haptic interfaces in the TAO architecture. Then, wrapper classes are provided for these TAO classes to encapsulate the network functionality. Meanwhile, the wrapper classes inherit the non-networked GiPSi classes and consequently have the same invocation interface as the GiPSi classes. Hence, the network implementation is transparently incorporated into the wrapper classes. Moreover, the resulting networked framework inherits from GiPSi its flexibility and potential for extensibility to a variety of surgical simulations.

More specifically, an example is given for the implementation of a remote haptic interface. Given a non-networked GiPSi HapticInterface class, we first define the non-primitive data types of parameters and return values of HapticInterface methods using TAO IDL (Interface Definition Language), for which TAO knows how to marshal to or demarshal from a data format suitable for network transfer. Meanwhile, we also define in IDL a TAOHapticInterface which takes parameters and returns value in the previously defined TAO types. The TAOHapticInterface object serves as a common object and resides at the haptic client side. It can accept remote method invocations (e.g., *ReadConfiguratioin*, *UseLowerOrderLinearModel*) from the simulation kernel. These remote

requests are eventually delegated to a private `HapticInterface` object of the `TAOHapticInterface`. On the simulation kernel side, a wrapper class `HapticInterfaceProxy` inherits `HapticInterface` and hides the details of `TAOHapticInterface`. `HapticInterfaceProxy` has a private member to hold the object reference to a `TAOHapticInterface` object at the haptic client side and, when the simulation kernel issues a method invocation on `HapticInterfaceProxy`, the invocation will be transformed into a method invocation on the `TAOHapticInterface` object. From the invocation interface point of view, the `HapticInterfaceProxy` class only differs from its base `HapticInterface` in the constructor, where some parameters are required for the TAO implementation (e.g., ORB initialization, remote object reference resolving, etc.). Therefore, GiPSi only needs a minimum of changes when extended to network environments through GiPSiNet.

While an appropriate use of TAO greatly simplifies the remote method invocations in GiPSiNet, TAO itself is far from being perfect. On one hand, TAO is a work in progress project and has a few unsupported Real-Time CORBA features. On the other hand, TAO can be better suited for real-time systems by, for example, adding real-time inter-orb protocol support such as the UDP-based RTP/RTCP. In real-time applications such as tele-haptics, timely data delivery is more important than reliable data delivery and UDP-based protocol such as RTP/RTCP are better suited in this case. RTP is a protocol providing support (e.g., timestamps) for real-time applications, and can be used for media-on-demand, interactive services such as VoIP, interactive distributed simulations, etc. RTCP provides feedbacks about the quality of data transmission, and therefore both the sender and receiver can react to the network dynamics according to the transmission quality reports.

However, UDP-based protocol can be subject to data losses, out of order data arrival, etc. Therefore, GiPSiNet adopts the following methods to compensate for the lack of network QoS.

*Compression.* GiPSiNet can benefit from a short data unit because short data (i) require fewer packets and in turn decreases the overhead of keeping track of multiple segments; (ii) decreases the loss probability; and (iii) has a shorter transmission time given the same network bandwidth. Therefore, GiPSiNet compresses data before they are sent and the algorithm is based on the standard Burrows-Wheeler algorithm [2].

*Retransmission.* The client resends data to the server at regular intervals, irrespective of whether the server has received the data or not. Thus, regular retransmission by the client can support timely data delivery to some extent.

*Forward Error Correction (FEC).* With FEC, a  $n$ -segment data unit is encoded into  $m > n$  segment, such that the reception of  $n$  segments out of  $m$  would enable the reconstruction of the data unit. Therefore, FEC can significantly reduce the probability of data losses and is therefore included in GiPSiNet.

*Adaptive Playout.* The network delay changes over time due to varying network conditions. This variance of network delay (jitter) disturbs the temporal relationships in the visualization and haptic media stream. To address this problem, GiPSiNet buffers the received packets and delay their playout, and sets the buffering time interval dynamically according to the network conditions and the server computation time.

## 4 Conclusion and future work

In this paper, we described the design and techniques of the GiPSiNet middleware project, which extends GiPSi to network environments and enhances the quality of networked surgical virtual simulations. In the future, we will deploy GiPSiNet and run experiments for a better understanding of the interaction between network and virtual environments. We will also build a benchmark suite of standard surgical tasks and perform expert reviews to systematically evaluate the qualitative and quantitative performance of GiPSiNet.

## Acknowledgement

The authors gratefully acknowledge Technology Opportunities Program (TOP) of Department of Commerce, NASA NNC05CB20C, NSF CCR-039910, and the Virtual Worlds Laboratory at Case Western Reserve University.

## References

- [1] <http://www.cs.wustl.edu/~schmidt/tao.html>.
- [2] M. Burrows and D. J. Wheeler. A block-sorting lossless data compression algorithm. Technical Report 124, Systems Research Center, Digital, 1994.
- [3] T. Goktekin, M. C. Cavusoglu, F. Tendick, and S. S. Sastry. Gipsi: A draft open source/open architecture software development framework for surgical simulation. In *International Symposium on Medical Simulation*, pages 240–248, 2004.
- [4] M. Henning and S. Vinoski. *Advanced CORBA Programming with C++*. Addison Wesley, 1999.
- [5] V. Liberatore, M. Cenk Cavusoglu, and Q. Cai. Gipsinet: An open source/open architecture network middleware for surgical simulations. In *14th annual Medicine Meets Virtual Reality (MMVR) conference*, 2006.
- [6] J. Smed, T. Kaukoranta, and H. Hakonen. Aspects of networking in multiplayer computer games. In *Proceedings of the International Conference on Applications and Development of Computer Games in the 21st Century*, pages 74–81, 2001.
- [7] F. Tendick, M. Downes, T. Goktekin, M. C. Çavuşoğlu, D. Feygin, X. Wu, R. Eyal, M. Hegarty, and L. W. Way. A virtual environment testbed for training laparoscopic surgical skills. *Presence*, 9(3):236–255, June 2000.