

# Use of Network Modeling Tools in the Nile System Design

Paul Avery

*Department of Physics, University of Florida,  
Gainesville, FL, 32611 USA*

I discuss the relevance of network modeling tools to high energy physics experiments, particularly for understanding data flows and identifying possible bottlenecks during data analysis. The Nile collaboration is using these tools to model a fault-tolerant computing system distributed over geographically dispersed sites. I summarize Nile's evaluation of three commercial tools and our initial experience with OpNet Modeler from Mil3, Inc. These tools seem to offer an effective way of quantifying data movements from online acquisition to physics analysis, including how they are affected by storage mechanisms, distribution, and the characteristics of local and wide area networks.

## 1 Introduction to Nile

The data volume generated by modern HEP experiments, and its imminent growth because of planned upgrades, defines the scale of our computing problem more than any other single factor. Within the next few years both  $e^+e^-$  and hadron collider experiments will be writing data at rates ranging from 10–200 TB per year. Projected data rates at the Large Hadron Collider (LHC) are larger by still another order of magnitude.

The Nile project<sup>1</sup> (*Distributed Computing and Databases for High Energy Physics*, NSF National Challenge Award ASC-9318151) represents one attempt to manage the computing burden imposed by such large datasets. Nile is a collaboration of high energy physicists and computer scientists who are building a robust system for wide-area access to computing and data resources. We are designing a computer architecture and database system which would be scalable to current and future hadron collider experiments, with the initial implementation focus on the CLEO experiment. The project's name reflects its goals of – I can't believe I'm saying this – “transporting information to distant sites and making villages of computing resources available to all its users”.<sup>2</sup>

CLEO's current and projected data storage needs are similar to those of other HEP experiments. In four years it will have accumulated roughly 100 TB of data on tape, with approximately 1–2 TB cached to disk to provide quick access to the most frequently used information. Many dozens of users, local and remote, using heterogeneous computing hardware, and having a variety of usage patterns, will sift through this data many times. A total of  $2 - 5 \times 10^4$  SpecInts of computing power will be required during this time.

Our goal within Nile is to provide a *fault-tolerant*, distributed software system which can handle this more than one order of magnitude increase in computing needed by CLEO, provide *efficient* and *transparent* access by all users, local and remote, to all data, and still be scalable by another two orders of magnitude to be relevant to the computing needs of the LHC era. These are, to say the least, ambitious goals. The first major milestone is a “Fast-Track” system which will

provide distributed computing access to present datasets using the current data model.<sup>2</sup> Later versions will provide a more database oriented data scheme<sup>3</sup> with a sophisticated architecture which will allow jobs to migrate to other computer centers, with the results automatically collated and sent back to the user.<sup>4</sup>

## 2 Data Modeling

### 2.1 *The Need for Data Modeling*

Data storage and retrieval is a major concern in Nile because the distributed nature of computing resources complicates access. It was felt that we might be better able to predict network behavior by using commercial modeling tools, particularly those that have built-in support for dataflows and networks.

The use of sophisticated Monte Carlo models to mimic physics processes and the detector response to these process is certainly known to High Energy Physics. However, modeling how data are collected and analyzed has never been a mainstream activity the way it has in other scientific and commercial endeavors. The case for modeling seems straightforward: (1) one can run simulated experiments on protocol changes, data storage and distribution, and the overall architecture; (2) modeling provides a “cheap” way to study changes to the architecture and data distribution since no hardware is affected; and (3) modeling reduces guesswork since issues can be studied in depth before any commitment is made. Specific questions are discussed below.

### 2.2 *Issues to be Addressed*

A distributed computing system presents many challenges to efficient access to data by a large user base. The following is a short summary of issues that we want to address.

*Data bandwidth* is an important measure of performance. How will limited Wide Area Network (WAN) bandwidth limit overall performance and what constraints will it set on the architecture? Are a few fast file servers better than many distributed servers? What are the tradeoffs involved in recomputing physics quantities vs. accessing remotely stored data? How will additional network components such as Ethernet switches, Fast Ethernet segments and switches and ATM networks affect bandwidth?

*Network protocols* need to be studied. How do current protocols limit throughput on LAN/WAN segments? Is CPU load during network transfer an important consideration? Will changes to more lightweight protocols make a significant difference?

*Data storage and distribution* issues may prove to be the most difficult to resolve. Where and when should data be replicated? Is there an optimal number of sites? Is there an efficient algorithm for deciding when data should be sent across networks? How should user data be handled? How do data flow in hierarchical systems and how much caching should be used at all levels of such a scheme?

### 3 Commercial Tools

#### 3.1 *Brief Comparison of Tools*

We considered three commercial packages:<sup>5</sup> OpNet Modeler, BONEs Designer and Comnet III. These tools are not inexpensive: typical list prices run from US\$40K–\$140K. However, very good academic discounts are available, ranging from 90 – 99%.

Comnet III turned out to be a basic simulator: it did not, in our opinion, have sufficient support for advanced networking features, and we did not consider it further.

BONEs's strongest features are its graphical editor and excellent model library containing over 300 parts, including multiprotocol routers and support for Ethernet, FDDI and ATM protocols. It is based on a single modeling interface: all objects are represented as blocks and interact via an explicit object-oriented paradigm. This seems appropriate for modeling objects such as routers, nodes, etc., but we felt that it was clumsy for modeling the processes running inside the blocks. BONEs lacks a built-in model for noisy lines or dropped packets and runs somewhat slowly since everything is interpreted. It does have excellent manuals and technical phone support. Academic use licenses for BONEs Designer can be obtained for less than US\$10K.

OpNet has a hierarchical model definition supporting multiple views: Network, Node and Process, each supported by a graphical editor which allows the user to place objects in a simple and intuitive way. The Network level defines the overall topology for a wide area network, including geographical overlays showing site locations. It is defined through an object-oriented paradigm. The intermediate Node level defines data storage, node locations, software protocol interfaces (such as TCP/IP) and is also object based. The Process level stands out because it models the processes executing inside the nodes using state transition diagrams. Unlike the upper two levels, which are interpreted, processes are written in PROTO-C, a C-like language which is translated into C and compiled for quick execution (it is simple to configure your own favorite text editor to edit the code). The use of PROTO-C enables quite general models to be constructed and is probably OpNet's most useful feature. OpNet modeler can be obtained academically for under US\$1K per year, which does not include phone support. However, technical support is available by e-mail and fax and its manuals are of high quality.

Opnet also supports useful features such as noisy lines and dropped packets and has very good ATM support. On the downside, it lacks the ability to include CPU load effects on network performance and supports fewer commercial systems than BONEs Designer. The pricing and feature list supported by OpNet are very strong, however, and we decided to adopt it.

### 4 Some Experience with OpNet Modeler

The OpNet screen during execution can be seen in Figure 1. Several features about its operation should be noted. First, Opnet is X-based and manages multiple windows only inside its screen boundary using a non-Motif window manager. While the



transfer rate in the “fast” scenario was actually lower than in the “slow” one, a result shown by subsequent analysis to be due to packet loss resulting from a very high collision rate.

The system was modeled in OpNet and the node level editor was invoked to analyze the ethernet CSMA/CD mechanism. From the model we were able to make an educated guess that a timing parameter in the NIC might be the culprit. The problem appeared to be that the client kept sending packets to the server without waiting sufficient time for the server acknowledgment, leading to a phenomenon known as “ethernet capture”, wherein a node gains control of the ethernet and prevents access by other nodes. We were able to take the built-in ethernet model and change it to add a “politeness” factor, which is just a short delay added to the time between packet sends. This turned out to be the critical parameter since a small change to it had a dramatic effect on the simulated collision rate.

## 6 Future Directions

In spite of our limited success with OpNet, we are clearly on the rock-climbing part of the learning curve. We plan to use OpNet in the near future to study network throughput in our ethernet switch using parameters supplied by the vendor. The issues here are how the switch behaves under high load conditions with multiple segments simultaneously active. In the longer term we want to understand hierarchical storage models, particularly caching algorithms and storage and architecture strategies.

## Acknowledgments

I would like to thank my Nile colleagues: Allesandro Amoroso, Michael Athanas, Ken Birman, David Cassel, Andrew Caulkins, Chris Hawblitzel, George Hoffman, Ray Helmke, Wade Hong, Theodore Johnson, David Kreinick, Keith Marzullo, Michael Ogg, Aleta Ricciardi, Dan Riley, Mark Rondinaro and Eric Rothfus. I also wish to thank the National Science Foundation, National Challenge Project (award 9318151).

## References

1. See the Nile home page at <http://www.nile.utexas.edu>.
2. See talk by M. Athanas, *The Nile Fast-Track Implementation: Fault-Tolerant Parallel Processing of Lagacy CLEO Data*, in these Proceedings.
3. See talk by M. Ogg, *The Nile System Data Model*, in these Proceedings.
4. See talk by A. Ricciardi, *The Nile System Architecture: Fault-Tolerant, Wide-Area Access to Computing and Data Resources*, in these Proceedings.
5. OpNet Modeler from Mil3, Inc., 3400 International Dr. NW Washington, D.C., 20008; BONEs Designer from Alta Group of Cadence Design systems, Inc., 919 E. Hillsdale Blvd., Foster City, CA 94404; Comnet III from CACI Products Company, 3333 N. Torrey Pines Court, LaJolla, CA 92037.