OBS Simulation Tools: A Comparative Study

Vasco N. G. J. Soares[1,2,3], Iúri D. C. Veiga[2] and Joel J. P. C. Rodrigues[2,3]
[1] Superior School of Technology, Polytechnic Institute of Castelo Branco, Castelo Branco, Portugal
[2]Department of Informatics, University of Beira Interior, Covilhã, Portugal
[3]Institute of Telecommunications, Networks and Multimedia Group, Portugal
vasco_g_soares@est.ipcb.pt; iuridavid@gmail.com; joel@ubi.pt

Abstract — **Regarding network performance, simulators are the most frequently used instrument, in comparison with analytical tools and *in situ* measurements. Optical Burst Switching (OBS) is a new paradigm where the use of modeling and simulation tools is very important. This paper studies and compares the four most relevant simulators used in scientific and academic research to model OBS networks (Network Simulator 2 (ns-2), OBS-ns Simulator, NCTUns, and OBSsimulator). The study can be used either to assist choosing the most adequate simulator for a particular scenario or to help improving the existing ones (or even create a new simulator from scratch). It's important to identify the parameters and protocols not supported (or implemented) whose performance analysis on OBS networks is critical and needs to be studied. Available simulators have several limitations because either they are not appropriate because they are based on packet traffic instead of burst traffic, or make simplistic assumptions not taking into account all of the resource reservation protocols and parameters that can influence the overall OBS network performance. We conclude that, based on the objectives of a user, this paper furnish the guidelines helping to the best choice taking into account the available simulation tools for OBS networks.**

Index Terms — **Optical Burst Switching (OBS), Network Simulation Tools, Performance.**

I. Introduction

Bandwidth-hungry applications and services are driving the demand for increased transmission rates in optical networks. First-generation optical fiber optic networks are focused on solving the link bottleneck problem, but since they use optical fiber as transmission medium, and switching, processing and routing are done at the electronic level, this raises another problem called the "electronic bottleneck".

It is foreseen that future networks will be based on the Internet Protocol (IP), so the development of all-optical packet switches to solve the above problem is currently a point of much interest. However, optical packet switching technology is immature due essentially to the problems of synchronization, optical buffering and time to configure optical switch fabric. Optical circuit switching paradigm is an alternative, however, it is not suitable to Internet traffic characteristics, so the bandwidth would not be efficiently used [1]. Therefore, optical burst switching (OBS) appears as a technical compromise between optical circuit switching and optical packet switching, since it is a method for transporting traffic directly over a bufferless optical WDM network.

OBS uses out-of-band signaling where the control packet contains routing and scheduling information, and it is converted from optical level to electronic for processing and converted from electronic to optical level for delivering. The corresponding data burst goes with some delay (the offset time) and passes through the core nodes, previously configured, until the destination node. The network intelligence is concentrated at the edge of the network and the core nodes only have the responsibility to process the control packet (setup message), send it to the next node and configure the optical cross-connect (OXC) to switch the burst.

Although there are some OBS testbeds such as ATDnet, TBONE, or reported in [2], given the non existence of Optical Burst Switched networks in the real world, in order to allow the development, test and diagnose of network protocols stacks and services on OBS networks, researchers can use different tools that can be classified as analytical models, simulators or *in situ* measurements [3]. Analytical techniques are based on mathematical models and are useful at the beginning stages of projects. Simulation techniques allow the modeling in detail of the studied system and they are flexible since they allow the studying of its performance under various conditions. The obtained results are easier to analyze than experimental results because they are repeatable. Simulation is also economical because it is possible to carry out experiments without the actual hardware (which is the case of OBS). *In situ* measurements are an expensive, slow and not flexible technique that usually is only used at the final stage of a project to examine the projected systems robustness and performance.

The study of OBS networks is currently in a phase where there are still several possibilities to reflect on. Therefore, simulators are essential to evaluate how these possibilities can influence the network performance.

The main objective of this study is to compare the support of the most relevant simulation tools in scientific and academic research, for specific features of OBS networks, and to analyze other general capabilities. Taking into account the user objectives, this work will help to determine the most complete simulator, and to improve the existing ones (or even create a new one) by identifying key network parameters and protocols not yet supported whose performance analysis needs to be evaluated.

The remainder of this paper is organized as follows. Section II provides an overview on simulation and modeling concepts. This section defines the characteristics analyzed for each studied OBS simulation tool. Section III presents a comparative study of the simulators. Section IV provides conclusions and future work.

II. OBS Simulation Tools

OBS technology raises a number of questions related to network parameters such as the network size and topology, the number of channels per link, the number of edge nodes per core, the edge to core node delay, the propagation delay between core nodes, the wavelength conversion, the burst offset length, the setup message processing time and the optical switch configuration time. These parameters may have impact on the performance of the different resource reservation protocols, such as Just-In-Time (JIT) [4], JumpStart [5], Just-Enough-Time (JET) [1], JIT$^+$ [6], Enhanced JIT (E-JIT) [7], or Wavelength Routed OBS network (WR-OBS) [8], therefore on the overall network performance.

This impact can be evaluated through the use of simulation tools. Simulators solve many difficulties, namely the need to build a real system, conducting controlled and repeatable experiments (at beginning, they should not impose limits to experimental scenarios) and perform sensitivity analyses. Furthermore, they allow the reproduction of simulation results, which is the basis for scientific advance.

Different types of simulation can be done on networks, such as macroscopic, microscopic and emulation. Macroscopic network simulation such as the evaluation of flows in "pipes" requires a coarse-grain discrete-event simulation (abstraction of a system as a set of dependent actions and events) and possibly a mathematical simulation (based on equations). Microscopic network simulation is a packet-level simulation which involves a fine-grain discrete-event simulation. It is also possible to be less abstract and simulate how actual flows go through a network, and we may have emulation.

In order to evaluate the simulators, it was defined a set of properties under study, that are the following: generic features/capabilities, specific support for OBS networks, model building (graphical model construction, model building using programming/access to programmed modules, or experimental design), output analysis, animation (animation or real-time viewing), support/training (user support, user group or discussion area, training courses, onsite training, or consulting available), system requirements and finally license type. This deep analysis will help us to build the comparative study presented at Section III.

This paper discusses the following simulation tools: ns-2 [9], OBS-ns [10], NCTUns [11] and OBSsimulator. A brief overview on J-Sim, OPNET Modeler, MATLAB & Simulink, and OptSim, is also considered. Next subsections present each simulator in detail.

A. *Network Simulator 2 - ns-2*

Network Simulator version 2, widely known as "ns-2" is a simulator based on a project started in 1989 called Real Network Simulator. Nowadays, it is supported through Defense Advanced Research Projects Agency (DARPA) with Simulation Augmented by Measurement and Analysis for Networks (SAMAN) and through National Science Foundation (NSF) with Collaborative Simulation for Education and Research (CONSER), both in collaboration with other researchers including The ICSI Center for Internet Research (ICIR).

ns-2 is a discrete event simulator for networking research, which allows simulation of TCP, routing, and multicast protocols over wired and wireless (local and satellite) networks [9]. Natively, it does not have a specific support for OBS networks. In order to improve run time efficiency, this simulator uses C++ programming language to implement object models and event scheduler. User defines and configures network details such as topology, applications, traffic types, the start and end point of the simulation and other parameters, using MIT Object Tcl (OTcl) scripting language that does not need to be compiled. Graphical model construction is also available with the use of drag-and-drop graphical user interface (GUI) tools such as NS Workbench.

An ns-2 simulation generates a trace file that contains topology information and packet traces. Later it can be animated or analyzed. Nam - network animator [12] is an animation tool for viewing simulation traces, so real-time viewing is not available. ns-2 software package contains an optional component called xgraph, a plotting program that can be used to create graphic representations of simulation results. Support documentation is available to users and developers [13].

ns-2 is typically built and run under Linux/Unix environment, but it is also possible to use it under Microsoft Windows using Cygwin application. Xgraph component runs on Unix platforms with X windows, however, support for Windows is not available. It is widely-used for networking research and it is available in the public domain on [9]. It's most recent release is version 2.31.

B. *OBS-ns Simulator*

The OBS-ns simulator was released by DAWN Networking Research Lab from University of Maryland as an extension to the ns-2 simulator for OBS networks. Its latest version 0.9 is unstable and it has limitations to simulate OBS networks.

A redesigned version of DAWN Lab's simulator is the OIRC OBS-ns Simulator [10], a simulation tool developed by Optical Internet Research Center (OIRC) and Samsung Advanced Institute of Technology (SAIT). It´s main goals are to solve the problems of version 0.9 and improve the software, introducing new features.

As its predecessor, OIRC OBS-ns is an event-driven simulator built on the ns-2, so it inherits its properties like the fact that object models are also developed in C++ and OTcl is still used for specifying and configuring the network simulation environment.

As shown in [14], to run an OBS network simulation it is necessary to create an OTcl script that contains besides other elements not relevant in the context of this work, the network topology, where edge and core nodes are defined as well as duplex fiber links between end nodes with specific bandwidth per channel, propagation delay, number of control and data channels, and the total number of channels. It is also necessary to build a routing table using shortest path routing and to specify traffic streams (which is done in the same manner as in ns-2). There is also an extended list of OTcl parameters with default values that can be changed to specify details about the OBS network, such as the delay of optional fiber delay lines (FDL) at end of link, the size of burst header packet (BHP) and DB overhead, the burst timeout, the maximum burst size, the offset time for class, the burst control packet (BCP) processing time in core classifier, the BCP processing time in edge classifier, the delay of FDL used for scheduling, the number of FDLs in a node, the maximum number of FDLs used per node or per path, the electronic buffering option, and the edge node electronic buffer size.

Based on available documentation and our experiments, it was not possible to determine which resource reservation protocols are supported in this simulator.

OBS-ns simulation output is written to statistics and trace files which can be processed using whatever tools the user desires, but in order to facilitate the process of getting results from the raw data, this simulator offers the possibility to plot graphs that analyze the throughput, delay and loss rate. As shown in Fig. 1, simulation animation is also available using Nam animator.

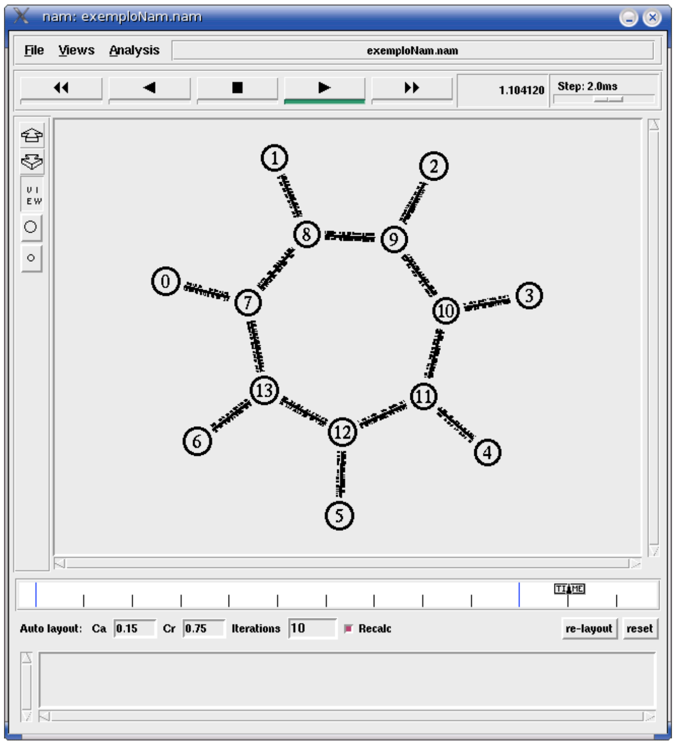

Fig. 1. Nam displaying OBS network simulation traces.

A full tutorial for this simulator can be consulted at [14]. OBS-ns can run under Linux/Unix or in Windows (using Cygwin). However, it needs ns-2.28 previously installed.

It can be downloaded on [10] with a copyright notice to University of Maryland, and an indication that this software is free for researching purposes and it is not available for commercial purposes.

C. NCTUns

NCTUns - National Chiao Tung University Network Simulator, belongs to SimReal Inc. a virtual company operated by the Network and System Laboratory (NSL), Department of Computer Science, National Chiao Tung University (NCTU), Taiwan. This simulation tool is both a network simulator and emulator that can be used for different purposes such as network-planning, research, application program performance evaluation and education [11]. It supports diverse network protocols/models/applications and types of network links and devices used in both wired and wireless IP networks, and it applies the event-driven method to its simulation engine. Its simulation engine has an open system architecture, it is open source, and it is implemented in the C++ language.

According to [15], it supports Optical Burst Switched Networks, since several modules are provided to construct the protocol stacks used in OBS networks, and OBS network devices can be used. It is possible to set the details of routers and switches, such as burst assembly (timeout, minimum burst length, maximum queue length), wavelength channel assignment (number of control and data channels), wavelength conversion, reservation scheme (only JET protocol is available), control packet processing time, contending control packet resolution method and contending burst resolution algorithm. The wavelength channels of an optical link are independent and can be individually configured, setting its bandwidth, signal propagation delay, bit-error-rate and down time periods.

Fig. 2 presents the GUI program that enables graphical model construction, allows drawing network topologies and to configure protocol modules. It also allows plotting network performance graphs and playback the animation of a logged packet transfer trace.

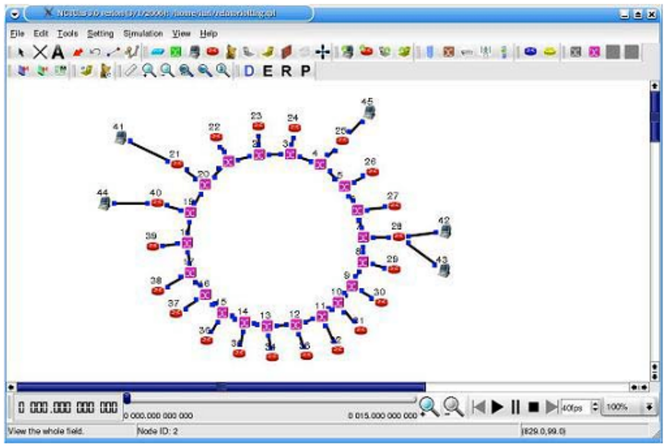

Fig. 2. OBS network with a ring topology represented on NCTUns.

User support is available at [16] on the form of manuals, papers and demo videos. NCTUns latest version 4.0 has been tested on Red-Hat Fedora 7 Linux operating system with specific kernel version 2.6.21, although other Linux distributions that use the same kernel may work. It is available for download on [11]. If NCTUns is used for research or study it is requested to cite [17] on papers and reports. Commercial licenses are also available.

D. *OBSsimulator*

Teng and Rouskas from North Caroline State University, USA, developed OBSsimulator, a tool that is able to model and simulate OBS networks, using C++ programming language. These authors gently ceded OBSsimulator to one of the authors of this work, and gave further indications on how to use it. Until this moment this tool has not been published, however, there are published results obtained by this simulator in [6].

Main objectives of OBSsimulator are the simulation of different resource reservation protocols based on the burst loss probability, to study the influence of different network profiles on the performance of OBS networks, including different network topologies.

OBSsimulator misses graphical model construction. To run a simulation it is necessary to create a file with a fixed name (*n1.net*), and using a special syntax and semantics. Its contents define the network topology, the scheduling algorithm, the number of edge nodes per core, the percentage of edge nodes that can generate bursts per core node, and the propagation delay between core nodes. Then the program is called on command line giving has option parameters the resource reservation protocol (JIT, JET, Horizon, JIT^{+} and JumpStart), the inter-arrival process (time between events), the burst generation ratio per node, the number of available data channels per link, the number of wavelength converters in each node, the time to process the setup message, the time to configure the optical cross-connect, and the propagation delay between edge and core nodes.

Simulation results are only presented on console so advanced data output analysis by the means of plotting graphs is not available. The same happens with animation or real-time viewing, which are not supported. OBSsimulator doesn't have documentation support, and it was only possible to determine that Linux is required to execute it. Since it is a work not published and created for a private use, it has no license type and is not available for download.

E. *Other simulation tools*

The main goal of this comparative study is to evaluate the capabilities of the most relevant simulators under use in research to model OBS networks. Furthermore, we extend this work making a short presentation of other alternative tools that have also been used.

J-Sim is a component-based, compositional simulation environment, and it is based on the autonomous component architecture. It does not include any specific component to model OBS networks. OPNET Modeler from OPNET Technologies, Inc. does not have specific modules to simulate OBS networks, however the capabilities of the model library are not limited so it is possible to develop protocols or device models with OPNET process and node editors. MATLAB from The MathWorks, Inc. can be used to develop an OBS simulation environment. Furthermore, its add-on called Simulink is a platform for multidomain simulation and model-based design for dynamic systems that provides an interactive graphical environment and a customizable set of block libraries, and can be extended for specialized applications. Finally, OptSim from RSoft Design Group, Inc. is another powerful simulator that does not include specific support for OBS networks.

III. Simulators Comparative Study

Although this paper presents the four more relevant simulation tools for researching on OBS, we restrict this final comparison to simulators that have specific support for OBS networks, which is not the case of ns-2. In order to compare OBS-ns, NCTUns and OBSsimulator, Table I presents a comparison on the following features: OBS resource reservation protocols supported, number of OBS network parameters considered, mechanism for network traffic generation, simulation model construction type, output analysis of the simulation results, animation or real-time viewing while the simulation is executing, user documentation available, learning curve, installation difficulty, programming language used to develop the tools, general system requirements, download availability, and license type.

TABLE I
Summary of Simulator Tools Features

	OBS-ns	NCTUns	OBSsimulator
OBS Protocols		JET	JIT, JET, Horizon, JIT+, JumpStart
OBS Parameters	14	13	11
Network Traffic Generation	Application code	Real-life protocol stacks (TCP/IP)	Application code
Model Building	Script	Graphical model construction	Script
Output Analysis	Plot graphs	Plot graphs	Console output
Animation or Real-time Viewing	Animation	Animation	Not available
Support/Training	Available	Available	Not available
Learning Curve	Steep	Short	Short
Installation Difficulty	Medium	Hard	Easy
Programming Language	C++	C++	C++
System Requirements	Linux	Linux	Linux
Download	Available	Available	Not available
License Type	Free for research, commercially not available	Free for research, commercial licenses available	Private use

Regarding OBS resource reservation protocols supported, OBSsimulator is the best tool since it includes the evaluation of more protocols. Although, it offers the possibility to study a smaller number of network parameters, a direct comparison is not suitable here since the presented simulators support different parameters. Regarding the network traffic generation it is important to notice the key trade-off between accuracy and speed since more abstract simulations are fastest to run, but less abstract ones are the most accurate. Most network simulators simulate real-life network protocol implementations with limited details [17]. This restricts their complexity but may lead to incorrect results. So, if possible, it is advisable to use real-life protocol stacks rather than their abstractions (this is the approach of NCTUns), but this will cause simulations to take more time to be finished.

Graphical model construction of the simulation is only available at NCTUns GUI. OBS-ns and OBSsimulator use script languages to create the network topology and configure the protocols and parameters to be used. OBS-ns and NCTUns offer the possibility to interpret simulation output results by the means of ploting graphs. Such option is not available on OBSsimulator. Its produced results are text based which reduces the perceived user-friendliness. None of the simulators allows real-time viewing animation of a simulation (while it is running), and only OBS-ns and NCTUns permit to view an animation after the simulation has been concluded. OBSsimulator lacks user support/training, unlike OBS-ns and NCTUns.

Given its GUI NCTUns has a short learning curve. Both OBS-ns and OBSsimulator use script languages, but the OTcl used in OBS-ns may step up the learning curve. The process of installing NCTUns is the most complex given the necessity of using a specific Linux distribution and kernel version, OBS-ns needs a previously installed version of ns-2, and OBSsimulator does not need to be installed since it consists on an executable file. All of the above tools have been developed in C++ and have as general system requirements Linux operating system. Finally, since OBSsimulator is a work not published and for private use only, it is not available for download and doesn't have a license. OBS-ns and NCTUns are available on the public domain and both are free for researching purposes.

In order to validate simulator results one against each other, several experiments on different network scenarios were executed. However, no conclusions can be taken given their different assumptions and support for OBS network model and parameters.

IV. Conclusions and Future Work

OBS has been considered an effective underlying technology for applications with large data requests and sensitive to path delay. It appears as a good solution to be the infrastructure of the next generation optical Internet. Giving the nonexistence of these networks in the real world and the current stage of their study, simulation is essential to evaluate the behavior, performance and to help to improve them.

We provided some insight on the most popular simulation tools used in scientific and academic research on OBS networks. We believe that this study can be helpful on picking the best of the existing simulation tools with OBS support for a particular scenario. But the major contribution is the conclusion that existing simulators either are not appropriate because they are based on packet traffic instead of burst traffic, or make simplistic assumptions not taking into account all of the resource reservation protocols and parameters that can influence the overall OBS network performance. Based on the objectives of a user, this paper furnish the guidelines helping to the best choice taking into account the available simulation tools for OBS networks.

OBSsimulator offers the broader support in respect to resource reservation protocols study. However, some work remains to be done, since it does not support any kind of animation or real time viewing of the simulation, and its user interface and output analysis are not user friendly.

The above work can be extended making the simulators validation, for example by verifying that each of the simulators implements accurately the OBS network model, devices and subsystems (entities). This is our interest for future work.

Acknowledgment

Part of this work has been supported by the Group of Networks and Multimedia of the Institute of Telecommunications – Covilhã Lab, Portugal, and by the Euro-FGI Network of Excellence of Sixth Framework Programme of EU.

References

[1] C. Qiao and M. Yoo, "Optical Burst Switching (OBS) - A New Paradigm for an Optical Internet", *Journal of High Speed Networks,* vol. 8, pp. 69-84, January 1999.

[2] I. Baldine, A. Bragg, G. Evans, M. Pratt, M. Singhai, D. Stevenson, and R. Uppalli, "JumpStart Deployments in Ultra High Performance Optical Networking Testbeds", in *IEEE Communications Magazine*. vol. 43, 2005, pp. S18-S25.

[3] P. Barford and L. H. Landweber, "Bench-style Network Research in an Internet Instance Laboratory", *ACM SIGCOMM Computer Communication Review,* vol. 33, July 2003.

[4] I. Baldine, G. Rouskas, H. Perros, and D. Stevenson, "JumpStart - A Just-In-Time Signaling Architecture for WDM Burst-Switched Networks", in *IEEE Communications Magazine*. vol. 40, 2002, pp. 82-89.

[5] A. H. Zaim, I. Baldine, M. Cassada, G. N. Rouskas, H. G. Perros, and D. Stevenson, "The JumpStart Just-In-Time Signaling Protocol: A Formal Description Using EFSM", *Optical Engineering,* vol. 42, pp. 568-585, February 2003.

[6] J. Teng and G. N. Rouskas, "A Detailed Analisys and Performance Comparison of Wavelength Reservation Schemes for Optical Burst Switched Networks", *Photonic Network Communications,* vol. 9, pp. 311-335, May 2005.

[7] J. J. P. C. Rodrigues, M. M. Freire, N. M. Garcia, and P. P. Monteiro, "Enhanced Just-in-Time: A New Resource Reservation Protocol for Optical Burst Switching Networks", in *ISCC'07 - IEEE Symposium on Computers and Communications*, Aveiro, Portugal, 2007.

[8] M. Düser and P. Bayvel, "Analysis of a dynamically wavelength-routed optical burst switched network architecture", *IEEE Journal of Lightwave Technology,* vol. 20, pp. 574-585, April 2002.

[9] "The Network Simulator - ns-2", http://www.isi.edu/nsnam/ns/, accessed at October, 2007.

[10] Optical Internet Research Center, "OIRC OBS-ns Simulator", http://wine.icu.ac.kr/~obsns/index.php, accessed at October, 2007.

[11] SimReal Inc., "NCTUns", http://nsl10.csie.nctu.edu.tw/, accessed at October, 2007.

[12] "Nam: Network Animator", http://www.isi.edu/nsnam/nam/, accessed at October, 2007.

[13] K. Fall and K. Varadhan, "The ns Manual (formerly ns Notes and Documentation)", http://www.isi.edu/nsnam/ns/doc/index.html, accessed at October, 2007.

[14] "Tutorial for Optical Burst Switch Networks in NS2", http://wine.icu.ac.kr/~obsns/docs.php, accessed at October, 2007.

[15] M. C. Yu, H. J. Tsai, C. Y. Huang, and S. Y. Wang, "Supporting Optical Network Simulations (OBS) on the NCTUns Network Simulator and Emulator", 2004.

[16] SimReal Inc., "NCTUns Support/Documentation web page", http://nsl10.csie.nctu.edu.tw/support/documentation/mainframe.htm, accessed at October, 2007.

[17] S. Y. Wang *et al.*, "The Design and Implementation of the NCTUns 1.0 Network Simulator", *Computer Networks,* vol. 42, pp. 175-197, June 2003.