Proceedings of the 2006 Winter Simulation Conference
L. F. Perrone, F. P. Wieland, J. Liu, B. G. Lawson, D. M. Nicol, and R. M. Fujimoto, eds.

SWORD: SCALABLE AND FLEXIBLE WORKLOAD GENERATOR FOR DISTRIBUTED DATA PROCESSING SYSTEMS

Kay S. Anderson*
Joseph P. Bigus
Eric Bouillet
Parijat Dube
Nagui Halim
Zhen Liu
Dimitrios Pendarakis

IBM T. J. Watson Research Center
19, Skyline Drive
Hawthorne, NY 10532, U.S.A.
*US Department of Defense.

ABSTRACT

Workload generation is commonly employed for performance characterization, testing and benchmarking of computer systems and networks. Workload generation typically aims at simulating or emulating traffic generated by different types of applications, protocols and activities, such as web browsing, email, chat, as well as stream multimedia traffic. We present a Scalable WORkloaD generator (SWORD) that we have developed for the testing and benchmarking of high-volume data processing systems. The tool is not only scalable but is also flexible and extensible allowing the generation of workload of a variety of types of applications and of contents.

1 INTRODUCTION

The wealth of novel Internet applications deployed today, such as web-browsing, chat, VoIP and IPTV offer unprecedented opportunities for enhancing our collaborative environment by enabling us to exchange, organize and search through vast amounts of information. These opportunities also inevitably create new challenges in harnessing the capabilities of these applications while protecting from their misuses: service providers must size their hardware and middleware appropriately in order to guarantee SLAs; IT departments must take proactive measures to ensure that their network is secure against intrusions and fraudulent activities; institutions must protect sensitive information (financial, medical, intellectual property) from advertently or inadvertently being leaked out and illegal or copyrighted materials from being leaked in; email users want to automatically and safely filter out unwanted emails before they reach their mailboxes. It is thus becoming crucial that we develop, in parallel to these applications, systems and tools that enable us to understand and make the most efficient and secure use of these applications. This, in turn, requires systematic methods for measuring the effectiveness of these tools in test-bed environments, and the ability to simulate information exchange resulting from the interactions of potentially millions of individuals in ways that are statistically and semantically realistic, reproducible and controllable.

There has been some prior work on devising benchmarking tools for emulating a collaborative environment. We envision that successful benchmarking of these approaches relies on testing them over diverse scenarios, incorporating myriad of users and/or machines, each of them characterized by different attributes resulting in a rich variety of data characteristics, different type of explicit or implicit semantic information (keywords, topics) and represented by different underlying social networks, which delineate message content and exchange behaviors between the participants. One approach for testing and benchmarking such systems and algorithms is to use prerecorded logs, such as combinations of chat logs, intrusion forensics, and VoIP records from pilot programs, which are publicly available on various internet sites. However relying only on these logs for testing the algorithms might be too restrictive and limited to the nature of information and embedded correlations present in these logs. Furthermore, this approach prevents testing of algorithms under activities which have not been observed earlier. Another approach is to develop parametric models for the behavior of users in interactive collaborative environments over the Internet using statistical

1-4244-0501-7/06/$20.00 ©2006 IEEE

studies of these web logs. Once these models are developed and well understood, we can use them to simulate information exchange data for different hypothetical scenarios with prescribed spatio-temporal correlations.

A critical challenge with the second approach is that it requires both *volumetric* and *contextual* statistics associated with the application of interest for efficient and accurate modeling. Volumetric statistics define the length of a session, fraction of time spent in different session states, transition probabilities between different session states, packet size distributions, etc. Contextual statistics define the topics and language used, marginal and/or conditional distributions of words or message lengths, vocal features (VoIP), etc.

In this paper we present a tool that addresses these challenges by providing a flexible and extensible platform for generating a wide range of workload types with both volumetric and contextual correlations.

The paper is organized as follows. Sec. 2 provides a background on the requirements of an ideal workload generator for distributed data processing systems dealing with high volume, contextually rich data and the challenges to develop such a tool. The section also discusses available tools and their limitations in meeting those requirements. Sec. 3 introduces SWORD and provide an overview of its architecture and run-time environment highlighting our key design choices. Sec. 4 describes our approach for modeling and representation of contents in SWORD. Sec. 5 talks about SWORD data factories and their APIs. Performance results on the scalability of SWORD are provided in Sec. 6. Finally we conclude in Sec. 7 with our ongoing and future works.

2 WORKLOAD GENERATION: CHALLENGES AND EXISTING TOOLS

We next discuss critical requirements for an ideal workload generator for testing data processing systems, which, in particular, processes, analyses, and makes intelligent deductions from high-volume, continuous, multi-modal stream data in a highly resource constrained environment.

2.1 Requirements and Challenges

Scalability: A crucial requirement is to generate very high traffic rates (of the order of Gbps) with relatively small amount of hardware resources to effectively test resource constrained systems dealing with large volumes of internet traffic. The difficulty is to reach a tradeoff that achieves the desired amount of background traffic while maintaining the adequate level of realism. For instance, we need to determine the optimum distribution of resources spent on replaying prerecorded traffic, synthesizing content, and performing real-time client-server transactions (i.e., stress-test a server application using a client agent under the control of the workload generator). These tradeoffs have to be empirically determined according to available CPU speeds, RAM sizes, storage capacities, storage transfer rates, programming languages used, and the objective of the simulation.

Content Rich: For testing tools that process content for filtering or indexation (as for spam-filters and search engines), we must be able to also produce a stream of coherent data content that is statistically and semantically realistic. The content may be topic-based, with possible correlations within streams. The content may further undergo multiple stages of real-time signal processing, such as noise insertion (static, background sounds), encoding (GSM, MPEG, MP3), and encryption.

Versatility of applications and protocols: There are literally hundreds of protocols, and new protocols are regularly created. As a consequence the workload generator must be modular, allowing existing protocols to be added as needed and new protocols to be added later on. The spectrum of protocols and applications representation should be tunable so that it can be offered in realistic proportions, with the ability to produce statistically realistic flow dynamics and traffic volumetrics, at user, application and transport level.

Specification and Insertion of Challenges: A challenge is a relatively subtle data compound or artifact that is explicitly concealed in the background traffic in order to recreate situations in which a given hypothesis is known to be true. It typically reflects the temporal and spatial correlation of multiple data sources interacting according to a prescribed pattern. It may also be a trigger that causes a component to break or introduces a change that the tested application must automatically detect and react in order to stress its component designs and autonomic aspects. The purpose of a challenge is to:

- Test the system's ability to accurately corroborate hypotheses in general, i.e., detect true positives versus false negatives, and also to test its ability to adapt to system/architecture perturbations. This is of concern to the algorithm developers who desire to verify the performance of particular algorithms and strategy embedded within the tested application. For instance, in order to test the analytics of an information processing system for intrusion detection, the challenge may consist of coordinated activities, such as Http, telnet, ftp, which appear innocuous when observed independently, but exhibit the hallmark of an exploit when viewed as a whole.
- The ability of the system architecture and components to support these strategies, the necessary system responses, and to handle the required volumes of data. It enables the system designers to stress and test the architecture and their design implementation. This type of challenge may consist,

for instance, of commands that shutdown part of the system in order to test its resiliency in the event of a system failure.

Challenges must mimic the background traffic to avoid detection based solely on obvious statistical dissimilarities. One of the difficulties is to be able to generate challenges that exhibit intricate, controllable, multi-source spatial and temporal correlations, having the same statistical properties as the background traffic. Or conversely to generate gigabits of background traffic that possess the statistical properties but not the particular correlations of the challenges in order to create background traffic to camouflage the challenges and thus stress the system's analytics.

Correlation and Synchronization of Contents: There is an increased need to test and benchmark stream processing systems that process a large number of continuous streams containing potentially correlated information. In particular we want to emulate the collective appearance of individuals - humans or machines - acting simultaneously. The resulting dynamics is a rich spectrum of interleaved networking activities. Analytics modules may process multiple streams to detect common patterns, interdependent events, content generated by common sources or related users. In testing such modules, the challenge is to generate a large number of streams, "inject" correlated test patterns across them and trace the ability of the tested system to successfully detect these correlations. Multiple types of correlation should be reproduced by such testing and traffic generation systems:

- *Contextual Correlations:* refers to the existence of *related* content (and hence challenges to the tested system) across different streams, possibly of different types.
- *Temporal Correlations:* corresponds to the appearance of related events or contents, separated by a time shift. Temporal correlations can appear both within the same stream (intra-stream) and across different streams (inter-stream).

Example of data generation that exhibit both contextual and temporal correlation is a sentence appearing in a chatroom, and a semantically related sentence which is later spoken in a different language in a VoIP conversation. Additionally, there is a need to capture correlations stemming from (stochastic) set relationships; for example to represent community of interests, with user being part of a group or a company or a company being subset of another company. More broadly, such testing tools need to support complex set relationships defined by social networks.

Flexible and Controllable Traffic Generation: In addition to the content model and statistical representation, which can be specified offline, the workload generator must also provide the ability to orchestrate the data generation and dynamically control the generated traffic both at volumetric and contextual levels. In particular it must provide tunable and dynamically controllable traffic intensity and traffic mix, and tunable content accuracy and challenge levels.

2.2 Existing Tools

The existing workload generation tools focus primarily on matching predetermined volumetric and timing properties and ignore statistical properties at the content level, such as content and contextual semantics. Most of the existing approaches for traffic generation are either application specific or lack scalability and/or modularity. Some candidate distributed workload generation tools of interest are LARIAT (Haines et al. 2001; Rossey et al. 2002), StreamGen (Mansour, Wolf and Schwan 2004) and D-ITG.

LARIAT developed by MIT Lincoln Laboratory and supported by DARPA is specifically optimized for intrusion detection (ID) evaluations. LARIAT uses models for usage patterns of different applications and use them for the synthesis of realistic user-sessions using real service and protocols. However, the emphasis is mostly on volumetric statistics of the generated traffic and not on the richness of the content. Further LARIAT lacks scalability (the best possible rate achievable is 100 Mb/s) and is also based on linux-systems. This may be a limitation for many commercial vendor applications and tools dealing with Windows supported internet applications. LARIAT is also costly to deploy (as it is all based on real-time sessions running over testbed) and there is no run-time control and tuning of the volumetric of the generated traffic. LARIAT also does not incorporate the ability for the users to specify scenarios and to specify and control cross-stream (multi-modal) correlations (volumetric, contextual and/or semantical) in the generated traffic.

StreamGen developed at Georgia Institute of Technology can be used to create a set of distributed services interacting via an application-level overlay. StreamGen is targeted at testing and benchmarking high performance computationally intensive distributed information-flow applications by generating the distributed computational and communication loads imposed by these applications. Since StreamGen is based on HttpPerf (Mosberger and Jin 1998), it only supports HTTP transport method. Further StreamGen lacks scalability, is linux-based and does not have the feature of real-time control of generated workload. Also there is no provision for content modeling in StreamGen and the generated traffic is mostly replay of prerecorded traces.

D-ITG developed at Univeristy of Napoli, Italy is a distributed multiplatform traffic generator for network testing and planning. D-ITG can generate multiple independent flows from a given traffic profile. However, the only tunable parameters for traffic profiling in D-ITG are the inter-packet

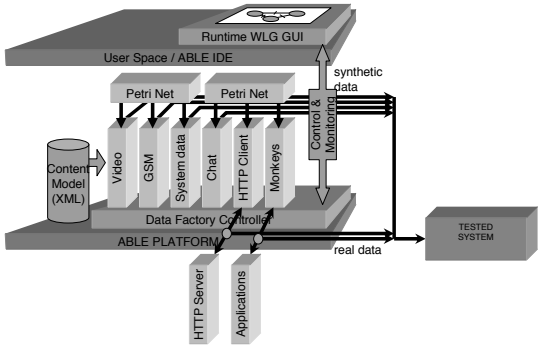

Figure 1: Architecture of SWORD

departure times and the packet sizes. Thus D-ITG lacks any content modeling, has no support to insert challenges in the generated traffic, cannot support inter or intra stream correlations, has poor scalability (about 600 Mb/s), and does not support run-time control and monitoring of the generated workload.

Some others available tools are WAGON (Liu, Niclausse, and Jalpa-Villanueva 2001), SURGE, GenSys, NetProbe, and Mercury LoadRunner. An indepth review of these tools revealed that the traffic generated by these tools is not suitable for testing and benchmarking systems that analyze data content and make intelligent decisions based on the content. We are not aware of any available commercial or academic tool that can meet all the requirements mentioned in Sec. 2.1.

3 SCALABLE WORKLOAD GENERATOR

SWORD is built on the Agent Building and Learning Environment (ABLE) platform, a Java IDE for the development and runtime deployment of distributed multi-agent systems. More precisely it extends the ABLE platform with a library of modular components implemented partially in Java for the ease of use and flexibility, and partially in C/C++ for efficiency, and specialized in the generation of particular types of data workloads. More information about ABLE is available at *http://ableinfo.userv.ibm.com*.

3.1 Architecture of SWORD

As depicted in Fig. 1, the main building blocks of SWORD are:

ABLE Platforms running on one or more machines. Each platform provides services for the management of multi-agent systems. SWORD leverages ABLE functionalities for agent management and monitoring, inter-agent communications, and optionally authentication and security. Furthermore, ABLE provides the developer with the ability to enable additional services such as service directory, and naming services. Through ABLE's agent lifecycle management functions, SWORD workload generation can be distributed on multiple platforms in order to achieve the desired level of scalability. The architecture enables the simultaneous generation of thousands of data streams using a mix of encoding types and transport protocols.

Data Factory Objects are the central components of SWORD. Data factory objects provide methods for translating the meta-data obtained from a content model described below into an actual data stream and package it into the appropriate networking protocol. Because they are typically CPU intensive, most of the data factories are implemented in C or C++ and are invoked from the ABLE platform agents through JNI. In addition to their content generation function, their API offers common control and monitoring capabilities such as specifying the content sources, or varying and reporting traffic volumetric. Data factories objects are instantiated and invoked via ABLE agents which execute in parallel, allowing the generation of simultaneous streams. SWORD provides a class of default data factory agents that repeatedly call the content generation function of their member data factory objects. In addition to default data factory agents, developers and more advanced users have the ability to quickly design complex ABLE data factory agents that coordinate and correlate the traffic flow of one or more data factory objects. A complex agent can for instance consist of a finite state machine that emulates an individual who accesse emails, browse the web, or participate in VoIP conversations. Such agents are represented as Petri-Net agents whose transitions e.g., activate email, http browser or VoIP data factories.

Data Factory Controller Agents are aggregators (or containers) of data factory agents. Their main function is to automate and hide the complexity of creating and of managing large populations of data factory agents from the user perspective. The Data Factory Controller API enables the user to specify the number of data factories in the pool managed by the controller and globally adjust the content generation parameters of the data factory agents in that pool. The user can configure the ABLE platform to contain several data factory controllers. It is, for instance, possible to organize the various data types (e.g., Audio, Video, HTTP, and Chat), into multiple data factory controllers so that the data factory agents of each type can be independently controlled and monitored.

Content Model are repositories that are globally accessible to all the data factories resident on the same ABLE platform. In addition they provide content sharing services allowing the distribution and access to the content across multiple platforms. The content model repository provides the ability to formulate the semantics of the content, and its statistical properties using a meta-data representation that is independent of the encoding and transport protocol used

to generate the data. For instance the content repository may contain an n-gram model for generating text content. An instant messaging data factory would use this model to generate instant messages, while a VoIP data factory would convert the randomly generated text into actual speech using a text-to-speech translation unit. The ability to share the same content model among multiple data factories, of possibly different modalities, enables contextual correlations across streams of different encoding and transport protocol types. The content model representation is discussed in details in Sec. 4.

3.2 Run-Time Environment of SWORD

SWORD consists of one or more distributed platforms, and one user interface to control and monitor the platforms remotely. The ABLE platform and SWORD user interface are written in Java and are thus supported by any platform that provides the Java 2 Runtime Environment, Standard Edition 1.4.2. However, SWORD components that generate the traffic usually invoke CPU intensive functions in the C/C++ space for efficiency reasons. These functions have currently been ported to Linux and Windows 2000/XP operating systems.

SWORD has limited CPU and physical memory capacity requirements that for most type of generated data are within what is currently available on the home computing market. The throughput of SWORD is a function of the CPU power, and its content richness a function of the physical memory. In particular, except for pre-recorded streaming content (video and audio), the content model is entirely cached in memory before starting the generator, and on hosts that have limited physical memory this design could result in an increased number of memory page swaps and poor performances. If necessary it is possible to distribute the contents to multiple off-the-shelves processing units in order to achieve the desired level of throughput and content richness.

4 SWORD CONTENT MODELING

In SWORD the content model is formulated in terms of decision trees. Nodes of the decision trees provide the logic for branching, and leafs provides the methods for generating the content meta-data which is then translated by the data factories into actual data streams. This classification between nodes and leafs is only a convenience for the content designer as a way to differentiate between the two behaviors. From a programming perspective, nodes and leafs all conform to a common set of Application Programming Interfaces (APIs) with various methods for getting the content and writing it into a buffer from where it can be processed by the data factory. Nodes transparently delegate this operation to one of their children until a leaf is reached. The programmer can easily extend the decision trees with new types of decision logics and meta-data types by deriving the new implementations from this common interface.

The content model is stored into an XML file, and is loaded into SWORD where it can be accessed by the data factories. A content model can contain multiple decision trees, each of which is given a unique name. The binding of the data factory to a specific decision tree is done during initialization of the data factory using the name of a decision tree provided in a configuration file or in the initialization method of the data factory. The binding is dynamically reconfigurable from a console GUI, allowing real-time scenario selections. The data factory uses iterators to pull data from the designated decision tree.

The iterator starts from a node in the tree and it then percolates down the tree from that node until it reaches a leaf, according to a decision path resulting from the logics of the traversed nodes. The iterator returns two data types into two separate buffers:

1. *Content meta-data* is generated when the iterator reaches a leaf of the tree. The returned meta-data is used by the data factory to create the actual payloads and protocol headers of the data streams.
2. *Content Annotation* is constructed while percolating down the tree. It provides the information on the decision path and how the meta-data was obtained, and can be used to package the data stream with a ground truth if desired.

For instance, as illustrated in Fig. 2 for VoIP, the content meta-data can be a text string and a speaker feature vector that are used to convert the text string into an audio stream with the speaker's voice. In the same example, the content annotation consists of contextual information indicating the name of the speaker, the mood and the topic of conversation. This annotation can be appended to the VoIP packets, or it can be sent out-of-band on a different channel, where it can be used as a ground truth to benchmark the analytics of the tested application.

Several data factories can simultaneously access the same decision tree. Because the state information used in making decisions, including the random number generators, are kept by the iterators, concurrent accesses are multi-thread safe without any performance penalty.

5 SWORD DATA FACTORIES

SWORD currently provides a set of data factories for Http traffic generation, emulated chat, GSM, MPEG-2 Transport streams, and meta-data, among others. All data factories conform to a common data factory java interface that provides the abstraction for modularity, and allows extension of the data factory set with new encodings and protocols.

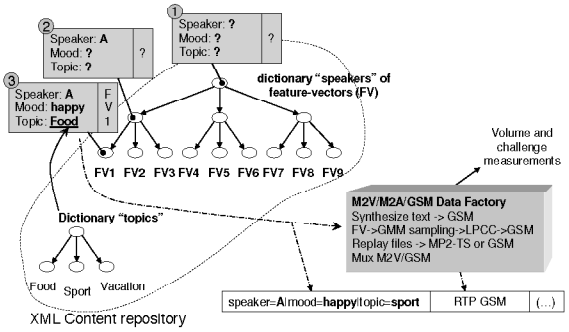

Figure 2: SWORD Content Model Representation and Use Case

The Data Factory public Java interface is shown in the following:

```
public interface AbleDataFactory{
  public void init
     (AbleDataFactoryControlAgentc,
      AbleContentRepositoryr, String url);
  public void init(AbleDataFactoryf);
  public void process(Stringcmd);
  public void process(String[] cmd);
  public long getVolume();
  public void resetVolume();
  public long getChallengeCount();
  public void resetChallengeCount();
  public void finalize();
}
```

The interface consists of initialization, process, monitoring, and finalization methods. The initialization method assigns a content model object and optionally a data factory controller to the data factory, and initializes the data factory with default values using a provided configuration URL. This information can also be copied from an existing data factory, allowing arbitrary data factory duplications. The process methods executes the data factory logic for transforming content meta-data randomly generated from the associated content model into a packet stream using the desired encoding and transport protocols. The arguments of the process method can be a string or an array of strings, the interpretation of which is dependent on the implementation of the underlying data factory object. The arguments includes the name of a decision tree in the content model from which the content is generated, the address(es) of target applications to which the content is sent, and protocol or encoding arguments that are particular to the data factory type. The process methods typically rely on native functions to perform CPU intensive tasks, and are thus optimized in order to minimize the overhead of traversing the JNI boundary.

Table 1: Scalability Results for SWORD

Data Type	% CPU	Rate	Bottleneck
HTTP	20%	1 Gb/s	NIC
GSM L-0	$< 1\%$	1 Gb/s	NIC
GSM L-1	100%	50 Mb/s	CPU
GSM L-2	100%	1.5-2 Mb/s	CPU
GSM L-3	100%	100 Kb/s	CPU
GSM L-4	$< 10\%$		Disk
Meta-data	20-25%	1 Gb/s	NIC

6 PERFORMANCE AND SCALABILITY

We have done performance studies of SWORD and the results have been very encouraging. The figures for SWORD's scalability tests done on IBM T-41 Thinkpad are shown in Table 1. Note that SWORD offers the capability for distributed workload generation. The measurements in Table 1 correspond to a single CPU and is not representative of the true potentials of SWORD for scalable workload generation. The scalability of SWORD is linear in the amount of CPU resources.

From the Table, we observe that HTTP and Meta-data workload generation is only limited by the network interface and with the same CPU, SWORD can generate about 5 Gb/s of HTTP or Meta-data.

For VoIP the generated rate depends on the level of content richness in the workload. Thus for Level-0 which corresponds to GSM packets with arbitrary contents, SWORD utilizes less than 1% of the CPU to generate 1 Gb/s of traffic. However, as the level of content richness in the generated workload increases (from Level-1 to Level-3) we observe that the maximum data rate with this CPU falls from 50 Mb/s (about 3500 simultaneous GSM streams at 14 Kb/s) to about 100 Kb/s. Note however, that at this level, SWORD is doing real-time text-to-speech synthesis using sophisticated tools. Further the generation rate of 1.5-2 Mb/s corresponds to about 100-140 simultaneous GSM streams.

7 ONGOING AND FUTURE WORK

One important area we are currently working is on the logging and benchmarking functionality of SWORD. There are several research challenges in this effort, including devising appropriate metrics for benchmarking, developing algorithms for comparing ground truth with analytics, and determining the level of aggregation in logging the data generated by SWORD for future validation. An additional area of extensions is the development of Petri-Net templates for different applications, such as chat, VoIP and web-browsing, that are customizable with scenario-specific parameters and can be used as building blocks for complex scenarios recreating real-world dynamics. SWORD user interface presents

several unique challenges. Currently, stream contents are modeled using XML files that provide great flexibility and versatility to the end user. We are working on a GUI that will facilitate the creation of these traffic content models without sacrificing the flexibility of the XML files. The trade-off between ease of use and versatility/increased functionality is a critical design consideration in this effort.

In terms of future research directions for SWORD, we are interested in both extending the coverage of data types and protocols as well as in providing enhanced customization, reconfiguration and robustness capabilities. Addition of new data types and protocols utilizes the extensibility and flexibility of the platform. However, this still requires workload characterization of new applications, protocols, and data types. Besides studying logs of such new workloads, we are interested in statistical tools that can help automate the process of template generation. Furthermore, we are interested in the ability to incorporate real-data sources and mix them with synthetic data in a transparent manner. Of critical importance is the ability of SWORD to simulate hardware/software failures inlcuding security failures, such as viruses, worms and DOS attacks. This feature will add to SWORD the ability to test a system's robustness against such failures.

REFERENCES

D-ITG (Distributed Internet Traffic Generator). Available via <http://www.grid.unina.it/software/ITG/index.php> [accessed July 17, 2006].

GenSyn. Available via <http://www.item.ntnu.no/~poulh/GenSyn/gensyn.html> [accessed July 17, 2006]

Haines, J. W., L. M. Rossey, R. P. Lippman, and R. K. Cunningham. 2001. Extending the DARPA Off-Line Intrusion Detection Evaluations. In *Proceedings of DARPA Information Survivability Conference and Exposition II*. Available via <http://http://www.ll.mit.edu/IST/ideval/pubs/2001/discex01_paper.pdf> [accessed July 17, 2006].

Liu, Z., Niclausse, N., and Jalpa-Villanueva, C. 2001. Traffic Model and Performance Evaluation of Web Servers. *Performance Evaluation* 46: 77–100.

Mansour, M., M. Wolf, and K. Schwan. 2004. StreamGen: A Workload Generation Tool for Distributed Information Flow Applications. In *Proceedings of International Conference on Parallel Processing (ICPP-04)*.

Mercury LoadRunner. Available via <http://www.mercury.com/us/products/performance-center/loadrunner/>[accessed July 17, 2006].

Mosberger, D., and T. Jin. 1998. httperf-A Tool for Measuring Web Server Performance. In *Proceedings of WISP* 59–67. Madison, WI: ACM. Available via <http://www.hpl.hp.com/research/linux/httperf/wisp98/httperf.pdf> [accessed July 17, 2006].

NetProbe. Available via <http://www.newobjects.com/downloads/NetProbe.zip> [accessed July 17, 2006].

Rossey, L. M., R. K. Cunningham, D. J. Fried, J. C. Rabek, R. P. Lippmann, J. W. Haines, and M. A. Zissman. 2002. LARIAT: Lincoln Adaptable Real-time Information Assurance Testbed. In *Proceedings of IEEE Aerospace Conference*. Available via <http://citeseer.ist.psu.edu/rossey01lariat.html> [accessed July 17, 2006].

SURGE. Available via <http://www.cs.bu.edu/~crovella/links.html> [accessed July 17, 2006].

AUTHOR BIOGRAPHIES

JOSEPH P. BIGUS is a Senior Technical Staff Member at the IBM T. J. Watson Research Center, where he is the project leader on the ABLE research project. Joe was an architect of the IBM Neural Network Utility and Intelligent Miner for Data products. He received his M.S. and Ph.D. degrees in computer science from Lehigh University and a B.S. degree in computer science from Villanova University. Dr. Bigus's current research interests include learning algorithms and intelligent agents, as well as multiagent teams and their applications to simulation and modeling, data mining, and decision support.

ERIC BOUILLET is currently at IBM T. J. Watson Research Center, NY, where he works on data modeling and test data generation. Before joining IBM, Dr. Bouillet has worked at Tellium Inc. from 2000-2004 and as a Member of Technical Staff in the Mathematical Sciences Research Center in Bell Labs/Lucent Technologies from 1998-2000. Eric holds an M.S. and a Ph.D. in electrical engineering from Columbia University. He also holds a joint degree from l'Ecole Nationale Suprieure des Tlcommunications ENST Paris and EURECOM Sophia Antipolis. His current research interests include data modeling and test data generation, design of optical networks and optimization of lightpath provisioning and fault restoration algorithms

PARIJAT DUBE received his M.S. in Electrical Communication Engg. from Indian Institute of Science, Bangalore in 2001 and his Ph.D. in Computer Science from University of Nice-Sophia Antipolis in 2002 where he was affiliated to INRIA, Sophia Antipolis, France. He joined IBM T. J. Watson Research Center, Hawthorne, New York in 2002. Parijat's

current research interests include stochastic modeling, distributed systems, computer networks, revenue management and pricing.

NAGUI HALIM is the Department Group Manager and the Head of Distributed Computing and Advanced Stream Processing Systems at IBM T. J. Watson Center, Hawthorne, New York.

ZHEN LIU received the Ph.D degree in Computer Science from the University of Orsay (Paris XI), France. He was with France Telecom R&D and then with INRIA (the French national research center on information and automation). He is now with IBM T. J. Watson Research Center and is the manager of the Next Generation Distributed Systems Department. Zhen Liu is a member of the IFIP W.G. 7.3 and an External Member of the Evaluation Committee of the Chinese Academy of Sciences. Zhen's current research interests are in distributed and networked systems, stream processing systems, sensor networks, performance modeling, distributed optimization and control.

DIMITRIOS PENDARAKIS is a Research Staff Member in the at the IBM T.J. Watson Research Center. His current research interests are in event-driven information systems, distributed system resiliency and autonomic computing. Dimitrios received his Diploma degree from the National Technical University of Athens, Greece and the M.S. and Ph.D. degrees from Columbia University, NY, NY. From 2000 to 2003 has was with Tellium, Inc., where he led the work on advanced control and management of intelligent, mesh optical networks. Dimitrios has taught a number of graduate level classes as an adjunct professor at Columbia University and Polytechnic University.