Modeling, Analysis, Measurement and Experimentation with the Tangram-II Integrated Environment *

Edmundo de S. e Silva
Ana P. C. da Silva
Antonio A. de A. Rocha
UFRJ, COPPE/PESC, IM/CSD
CxP 68511, Rio de Janeiro, RJ
21941-972, Brazil
edmundo@land.ufrj.br

Rosa M. M. Leão
Flávio P. Duarte
Fernando J. S. Filho
Guilherme D. G. Jaime
UFRJ, COPPE/PESC
CxP 68511, RJ, Brazil
rosam@land.ufrj.br

Richard R.Muntz
UCLA CS Departament
Boelter Hall,
Los Angeles, CA 90024, USA
muntz@cs.ucla.edu

ABSTRACT

A large number of performance evaluation tools have been developed over the years to support the analyst in the difficult task of model building. As systems increase in complexity, the need is critical for tools that are able to help the user throughout the whole modeling cycle, from model building to model solution and experimentation. In this work we describe the main features of the TANGRAM-II modeling environment. The tool has a powerful and flexible model interface, unique algorithms for the numerical solution of models, includes an event driven and fluid simulators that provides a variety of facilities useful for obtaining the measures of interest, and has a traffic engineering environment integrated with the other tool modules.

Categories and Subject Descriptors

C.4 [**Performance of Systems**]: Design studies, Measurement techniques, Modeling techniques, Reliability, availability, and serviceability; I.6.5 [**Simulation and Modeling**]: Model Development—*Modeling Methodologies*
; I.6.8 [**Simulation and Modeling**]: Types of Simulation—*Animation, Combined,Discrete event,Fluid*

General Terms

Measurement,Performance,Reliability

Keywords

analytical modeling, performance tool, simulation

1. INTRODUCTION

For more than 30 years performance evaluation tools have been developed to aid the modeling and analysis task. Some of these tools were tailored to specific applications, such as reliability/availability and queueing systems, and others allow the specification of general models (see [12] for a survey on the tools developed up to 1992). During these three decades user interfaces became sophisticated and there were significant advances in both analytical and simulation model solution techniques.

The user interfaces shifted from pure textual to graphical representations of the model components, often associated with a high level description language. Solution techniques have evolved allowing very large models to be analyzed, on the order of hundreds of thousands or millions of states. Steady state techniques benefited from the large amount of research on matrix analytical techniques [48, 40, 55], bounding techniques [9, 47, 36, 24], Kronecker algebra [34, 49] and those based on stochastic automata networks [53, 58]. Research on transient analysis advanced significantly [14], in particular methods based on the uniformization technique [26, 18]. In parallel, new simulation techniques have been developed such as those for handling rare events [30], including importance sampling and RESTART [60]. New techniques based on the concept of fluid flows [2, 32, 22] have also been employed successfully to lower the computational requirements of computer network simulation.

Many tools have been created over the years. For instance, Petri-Net and queueing network based tools [7, 8, 29, 37, 42, 20], to cite a few. With the growth of the Internet, the NS simulation tool [33] became very popular. Furthermore, the need for collecting measurements to understand the complex processes that compose Internet traffic also grew. As a consequence, a huge effort has been placed on the development of new measurement techniques capable of collecting a large variety of statistics useful to construct models targeted to traffic engineering (e.g. [52]). Needless to say, a variety of measurement tools is available.

In this paper we describe some of the features of the TANGRAM-II tool, that has been evolving for more than a decade. The main contributions of the tool are: the development of an integrated environment that includes a unique modeling paradigm for model especification, model solution using analytical solvers or simulators and, experimentation via traffic generators and active measurement techniques; novel analytical solution methods, and techniques for fluid simulation; new algorithms for active measurements. The applicability of such an environment and its modeling capabilities are shown through simple examples.

*This work is supported in part by grants from CNPq, CAPES and FAPERJ.

Valuetools'06 October 11-13, 2006, Pisa, Italy

In section 2 we present a general overview of TANGRAM-II. Section 3 introduces its modeling environment. Section 4 discusses a few analytical solvers of TANGRAM-II, as well as the simulators implemented. Section 5 presents the traffic engineering environment. Our concluding remarks and a summary of our contributions are presented in section 6.

2. AN OVERVIEW OF TANGRAM-II

The main purpose of a modeling tool is to provide the necessary support to the analyst to create an abstraction of the system under study and to help answer questions about the system behavior. First, the analyst has to estimate the range of values for the system parameters. These values can be acquired from measurements collected from real systems, via an experimentation laboratory setup, simply "guessed" from past experience, or by comparison against similar systems. Once the analyst has performed the necessary measurements, she can extract the parameter values from the measurements to feed the model, for instance, to obtain the distribution of a random variable that matches the collected data. The model is then constructed and solved via simulation or an analytical technique. The results obtained from the model undergo another analysis step to provide the answers to the questions faced by the analyst. The whole process is interactive in nature.

TANGRAM-II was built to help the analyst through the modeling steps. The tool integrates different environments for developing and analyzing computer and communication models. It was designed to support research, application development, and education in performance evaluation by providing the ability to construct a full range of simple to complex models and solve them by both analytical and simulation methods. The environment includes modules to conduct active measurements in a computer network and collect useful statistics to aid in parameterizing a model. TANGRAM-II allows the user to tailor the tool to specific application domains, such as queueing, dependablity, or network modeling. Besides solution methods available in the literature, we incorporated original techniques we developed aiming at providing a rich set of options to the modeller. These include techniques for transient analysis, and for solving a class of non-Markovian models [13, 4, 15]; algorithms for calculating measures useful for traffic analysis and experimentation [41, 19, 43] and those used for active measurements [56, 57]. The tool's fluid simulator has also distinct features from others that allows the construction of generic building blocks for different application domains. The same modeling paradigm is employed both for analytical and simulation models and is carried through the measurement and experimentation modules. The tool's modules were conceived to facilitate the addition of new techniques by providing textual interfaces among modules.

Figure 1 illustrates the main components of the TANGRAM-II environment. The top-level interface includes: (a) the model environment; (b) the traffic engineering environment and; (c) full fledge multimedia tools: a distributed whiteboard and the video/VoIP Freemeeting tool. The modeling environment supports the construction of simple to complex models, and their simulation or analytical (numerical) solution when possible. The traffic engineering environment provides the means to perform network measurements. The multimedia tools are useful not only for collaborative work but also serve as the basis for real network experimentation.

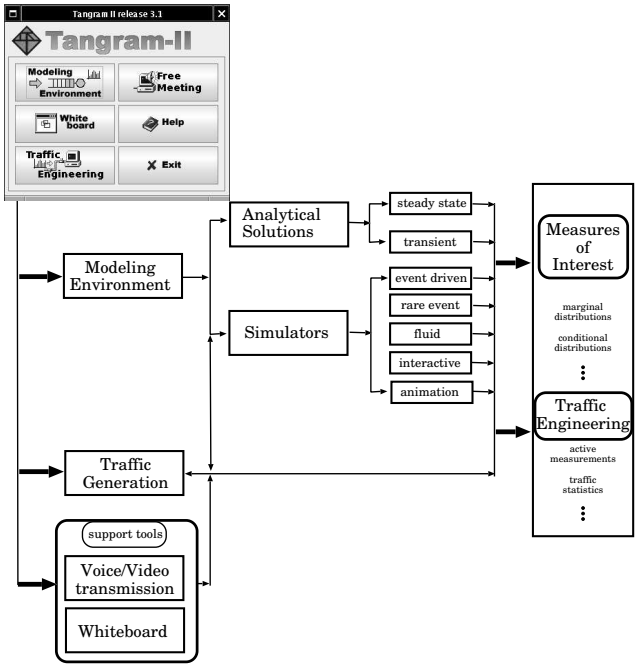

Figure 1: The TANGRAM-II environment.

3. THE MODELING ENVIRONMENT

The TANGRAM modeling paradigm was proposed in [3]. It was based on the observation that models are typically modular in that each consists of a set of *objects* that are connected in some form. Each object has an internal state that can change either due to an internally generated *event* or the receipt of a message generated by another object. Events are triggered spontaneously according to a given rate (and the associated distribution) provided that a set of conditions specified by the user are satisfied. An object can react to an internal event or received message by changing state and/or sending messages to other objects.

The tool includes a library of common objects that can be instantiated and connected to build a new model. New objects are developed using a template object that contains several attributes, such as: declarations, events, messages and initialization. The object's behavior is determined, following the tool's paradigm, from the actions that take place when one of the defined events occurs and how the object reacts when a message is received. The specification of the actions performed is done using a C-like language. The TANGRAM paradigm is quite powerful with respect to the ease with which one can specify objects with complex behavior. It allows the specification of general models and the construction of environments tailored to an specific application domain.

Each object has a (possibly empty) set of state variables whose values define its current state. In the TANGRAM paradigm, the overall model state is identified by the values of local object states. Other types of variables can also be defined for an object such as parameters, and constants. The state variables can change as the result of actions associated with events or a message that is received. Events occur spontaneously when the associated conditions are satisfied. The interval between events is a random variable with

a given distribution. When an event occurs, one of a set of actions is executed according to a given probability distribution. The actions can alter the current state of the object and/or result in one or more messages being sent to other objects that are connected to the sender. Objects react instantaneously to received messages and, as a consequence, new messages can be sent to other objects, and/or the state variables of the object may be altered.

Messages are sent to other objects via *ports*. Two or more ports are associated with a communication channel. Messages can be broadcast to all objects connected to a channel or directed to a specific object. The sender can include data in a message that is then evaluated by the receiver.

Figure 2 illustrates an example with two objects. The first is an aggregate of five Markov modulated on-off sources (thus the aggregate is a birth-death model). The packets generated by the sources are sent as messages to the second object, which represents a RED Active Queue Management queue, with exponentially distributed service times. RED discards arriving packets randomly, if the time-averaged queue length is above a given threshold. (We used the parameters suggested by [23].) In the model, the average queue length is quantized in order to obtain a discrete state space. This is an example of a Markovian model. For analytic solution the modeling environment can generate the corresponding transition rate matrix in numerical or in symbolic form, according to the parameters specified. The model can be solved analytically or by simulation.

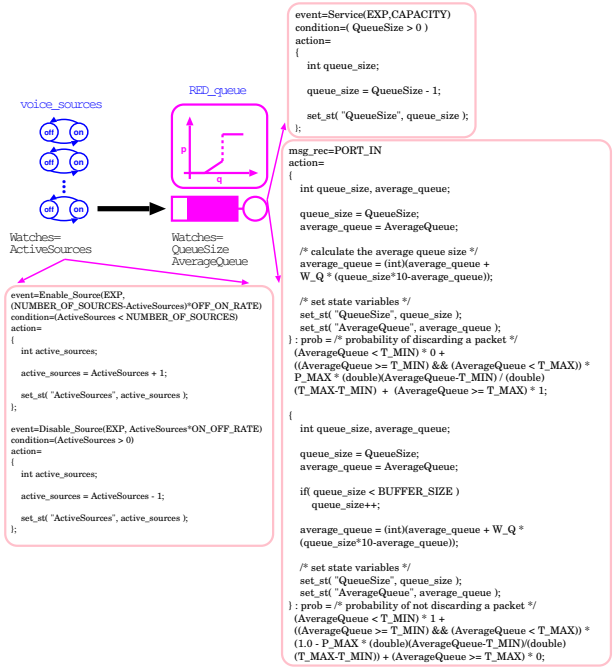

Figure 2: An example: model of the RED queue with 5 sources.

From an analytical model, measures of interest can be obtained as functions of the state variables. Among the measures of interest are: marginal distributions, distributions of functions of state variables, conditional distributions, etc. Another way to calculate measures of interest is via the reward attribute.

Each object may have associated with it a *reward* attribute. TANGRAM-II allows two types of rewards to be defined: a rate reward and an impulse reward. A rate reward has a given *name*, a set of *conditions* and associated *values*. Let $\mathcal{R}$ be a specified reward for an object, $\mathcal{S}_c$ be the set of object states that satisfy condition $\mathcal{C}$ associated with $\mathcal{R}$, and let $r_{\mathcal{S}_\mathcal{C}}$ be the corresponding value assigned to R. Then, $\mathcal{R}$ gains $r_{\mathcal{S}_\mathcal{C}}$ units of reward per time unit the object spends in the set $\mathcal{S}_\mathcal{C}$.

On the other hand, impulse rewards are associated with transitions in the model, and are useful to *count* events. Let $\mathcal{E}$ be an event of an object, p_i the probability of executing action i when $\mathcal{E}$ occurs and $\rho_{\mathcal{E},i}$ the impulse reward associated with the reward attribute $\mathcal{I}$. Then $\mathcal{I}$ gains $\rho_{\mathcal{E},i}$ reward units each time action i of event $\mathcal{E}$ is executed. Impulse rewards can also be associated with messages that are sent or received by an object.

The rewards described above are associated with an object and so the conditions and values have their scope limited to the set of state variables, events and messages of that object. However, one can also declare *global rewards*, and include the (time varying) cumulative value of the rewards just as any other variable in assignments to state variables, or in condition and action statements.

Rate and impulse rewards are a powerful approach to obtain measures of interest. One can specify complex conditions with rewards, and values that are constant expressions, dependent on state variables or on the cumulative values of other rewards. The simulators also make extensive use of the concept of rewards and rewards are the means by which the modeler specifies the measures to be calculated. As will be shown in section 4.2 reward and fluid models are tightly coupled concepts and the fluid simulator takes full advantage of the reward attributes. The way by which rewards are employed and the associated solution techniques is another contribution of the tool.

4. ANALYTICAL AND SIMULATION SOLUTION TECHNIQUES

In this section we focus on a subset of the solution techniques available in TANGRAM-II, including transient analysis methods and steady state methods that employ transient analysis as part of the solution.

4.1 Analytical Solutions

An extensive treatment of solution methods for Markov chains can be found in [58]. MARCA is a good example of a tool that implements a rich set of steady state solvers, from basic techniques to those appropriate for handling nearly completely decomposable (NCD) models and projection methods [58]. Major advances in the numerical solution techniques of Markov chains also include the matrix geometric and matrix analytic methods based on the seminal work of Neuts [48]. A tool that implements matrix-analytic algorithms and provides solutions for continuous and discrete time Markov chains of QBD, GI/M/1 and M/G/1-types is MAMSolver [54].

TANGRAM-II implements a few basic iterative methods for solving Markov chains such as *power*, *Gauss-Siedel* and *SOR*. The tool's direct method of choice is the so called GTH algorithm [27] (and its block version) which is a stable LU decomposition method for Markov chains based on

stochastic complementation [44]. This method works very well even for large state space cardinalities. Nevertheless, other methods could be easily coupled with the tool. Figure 3(a) shows the probability mass function for the RED model of Figure 2 (with a buffer size equal to 50), when the model is solved with GTH.

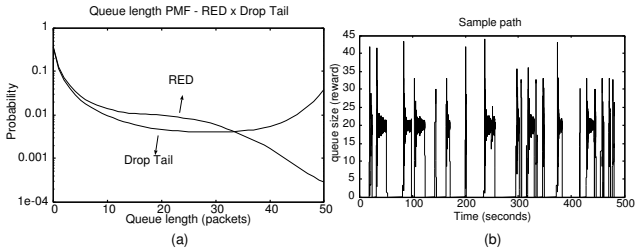

Figure 3: (a)PMF of the RED AQM queue using an analytical solver; (b) queue size using reward rate and simulation.

The computational complexity of some of the steady state solution techniques for Markov chains is dependent on the structure of the infinitesimal generator. For instance, if the infinitesimal generator is banded, then the GTH technique preserves this structure and its computational requirements are $O(Nk_lk_u)$ where N is the number of states in the model, k_u and k_l are the upper and lower bandwidth of the generator, respectively. The structure of the generator depends on the ordering of the states for the generator matrix. In TANGRAM-II the state ordering is determined by the order the states are searched during state exploration, but can be changed to correspond to any ordering of the state variables. In essence, given the ordering of the state variables, a state's numerical position is determined by the mixed radix number formed using its state variable values. Different orderings of the state variables, therefore result in different orderings of states and thereby the bandwidth of the resulting generator matrix. Thus, it is important for the analyst to visualize the generator matrix and be able to experiment with different state orderings, to choose an appropriate structure. TANGRAM-II has a module that allows the user to display the generated transition matrix as well as to permute the states according to the state variables in the model.

Figure 4 shows the state transition matrix that was generated for the RED AQM example of section 2. The left-hand side of the figure shows a state ordering with a large bandwidth. The right-hand size shows that the bandwidth can be reduced by a permutation of states based on a simple reordering of the state variables. As a consequence, GTH is more efficient when applied to the right-hand side matrix of Figure 4. This feature is particularly useful for teaching solution techniques. The tool also displays the transition probabilities between state pairs using a range of colors that allows the analyst to distinguish small from high probability values. Therefore, matrices with special structure can be discerned from others. The flexibility provided by the matrix visualization module is another contribution of the tool.

Transient analysis is relevant in practice when convergence to steady-state behavior is too slow to be of much value for engineering purposes. Consider a time homogeneous continuous-time Markov chain (CTMC) $\mathcal{X} = \{X(t), t \geq 0\}$ with finite state space $\mathcal{S} = \{s_i : i = 1, \ldots, N\}$ and infinitesimal generator $\mathbf{Q}$. One basic transient measure is the vector of state probabilities at time t, $\boldsymbol{\pi}(t) = [\pi_1(t), \ldots, \pi_N(t)]$ where $\pi_i(t) = P\{X(t) = s_i\}$. Several methods have been proposed to solve for $\boldsymbol{\pi}(t)$ (see [14] for a discussion of some of these methods).

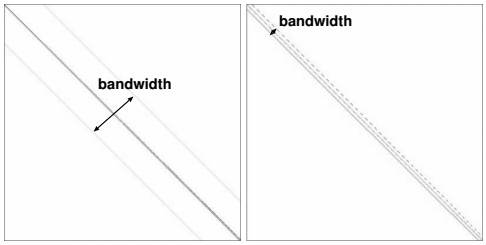

Figure 4: Two possible matrix structures for the RED AQM example of section 2.

Many transient measures of interest can be obtained by assigning reward rates to states or impulse rewards to transitions. As a consequence TANGRAM-II extensively uses rewards as a way to calculate useful measures. Let us assume that there are $K+1$ rate rewards $r_1 > r_2 > \cdots > r_{K+1}$, and state $s \in \mathcal{S}$ has a reward rate $r_{c(s)}$, where $c(s) \in \{1, \ldots, K+1\}$ gives the index of the reward corresponding to state s. A reward of $r_{c(s)}$ is accumulated per unit time in state s. The instantaneous reward rate at time t is the random variable $IR(t) = r_{c(X(t))}$, and the cumulative rate based reward during $(0, t)$ is $CR(t) = \int_0^t r_{c(X(\tau))} d\tau$.

Assume also that there are $\widehat{K}+1$ impulse rewards $\rho_1 > \rho_2 > \cdots > \rho_{\widehat{K}+1}$ associated with the transitions of the Markov chain. For any transition $s \to s'$, we let $\rho_{\widehat{c}(s,s')}$ be the impulse reward for that transition, where $\widehat{c}(s, s') \in \{1, \ldots, \widehat{K}+1\}$ is the index of the impulse reward corresponding to $s \to s'$. Let $N(t)$ be the number of transitions that occur during $(0, t)$. Then the cumulative impulse based reward during $(0, t)$ is $CI(t) = \sum_{n=1}^{N(t)} \rho_{\widehat{c}(\sigma_n)}$ where σ_n is the pair of states that represent the nth transition. As a simple example of a measure obtained from rewards, consider an availability model. The system availability can be obtained by assigning reward rates of $r_1 = 1$ and $r_2 = 0$ to the set of operational and failed states respectively. As a second example, consider a model where each state represents a given system structure, and associate a reward with each state according to the level of performance the system operates at while in that state. The cumulative reward $CR(t)$ is the so called system *performability* and captures the effect of a gracefully degradable system on its performance [45].

Since the early eighties, a technique called *uniformization* (also known as *randomization* or Jensen's method for calculating $\boldsymbol{\pi}(t)$ became widely used. (See [18, 26] and the references therein). Several reasons can be attributed to the popularity of the method, including its simplicity of implementation, numerical robustness and the probabilistic interpretation which was the key to many uniformization based algorithms that have been developed for calculating other transient measures of interest. Uniformization has even been proposed as an intermediate step to obtain steady state measures [18].

The method involves the transformation of a continuous-time Markov chain into a discrete-time analogue, and tran-

sient solutions are obtained by working with a problem in discrete time. Consider a continuous-time Markov chain $\mathcal{X}$ with finite state space $\mathcal{S}$ and infinitesimal generator $\mathbf{Q}$ with rate from s_i to s_j equal to $q_{i,j}$, and $q_i = \sum_{j \neq i} q_{ij}$ is the rate out of state s_i. Let $\Lambda \geq \max_i\{q_i\}$, and define the matrix $\mathbf{P} = \mathbf{I} + \mathbf{Q}/\Lambda$. Thus $\mathbf{P}$ is stochastic by choice of Λ, and since $\mathbf{\Pi}(t) = \mathrm{e}^{\mathbf{Q}t}$ is a solution of $\mathbf{\Pi}'(t) = \mathbf{\Pi}(t)\mathbf{Q}$, we have: $\boldsymbol{\pi}(t) = \sum_{n=0}^{\infty} \mathrm{e}^{-\Lambda t} \frac{(\Lambda t)^n}{n!} \boldsymbol{\pi}(n)$, where $\boldsymbol{\pi}(n) = \pi(0)\mathbf{P}^n$. This equation has several nice properties. It is simple, its evaluation involves the multiplication of positive numbers only, the error when the infinite series is truncated can be easily bounded, the sparseness of $\mathbf{P}$ is preserved. One difficulty was the potential overflow/underflow problem when calculating $\frac{(\Lambda t)^n}{n!}\boldsymbol{\pi}(n)$ for large n, but a simple algorithm exists to avoid such problem (see [12] which also points to the original work of the authors in 1986).

Extensions to the basic approach have been proposed to solve for models when $\mathbf{Q}$ changes at a finite number of time points or models with large Λt values. Using uniformization one can also perform sensitivity analysis of measures based on $\boldsymbol{\pi}(t)$ with respect to model parameters [31]. Furthermore, important measures from Markov reward models can be obtained. These include expected values such as: the expected reward at time t, $E[r_{X(t)}]$; the expected (rate) reward accumulated in $(0, t)$, $E[CR(t)]$; the expected reward obtained from transitions in $(0, t)$, $E[CI(t)]$; the expected time to achieve a given reward level. Although distributions are harder to obtain than expected values, considerable progress has been made in this area and the moments and distributions of the random variables above can be calculated using uniformization (see [18] and the references therein.)

Another class of applications for which uniformization has been employed is in calculating steady state measures for non-Markovian models. Roughly, embedded points are found such that the behavior of the system between these points is Markovian. Then the transient analysis of certain submodels (depending on the particular problem) are used to calculate the transition probability matrix between the embedded points. Application examples include the analysis of: the $GI/PH/1$ queue [25]; schedule maintenance policies for computer systems [11]; polling models with timeouts [16, 17]. In [15] a methodology was developed which addresses the solution of a broad class of non-Markovian models and associated measures. (See also [18, 14] for more details and references to related work.)

TANGRAM-II implements from basic to sophisticated transient analysis algorithms founded on uniformization. These include a variety of reward measures (both expected values and distributions), as well as solutions for a class of non-Markovian models as mentioned above, and approximation techniques (see, for instance, [13, 15, 4]).

In the next section we present the simulation capabilities that TANGRAM-II offers, and specially the fluid simulator. This facility, coupled with the analytical solvers for fluid models, creates a powerful environment for traffic engineering.

4.2 Simulation Techniques

A large variety of simulation tools are available. Perhaps the most widely used free/open code tool is NS [33]. The objective of TANGRAM-II is not to implement yet another simulator, but to provide a simulation environment well integrated with the other modules in the tool. For instance the user can construct a Markovian model, solve it with numerical techniques, change the distributions of certain events, simulate the modified model and compare the results. The modeling environment high-level interface is the same for both the analytical solvers and the simulator. However, the simulator includes additional high-level constructs to facilitate the modeling task. The simulator is also integrated with the traffic engineering environment. The user can import traces collected in that environment to be used as part of the simulation model. Traces can also be generated by the simulator for the traffic engineering module.

TANGRAM-II has a large set of simulation options. The user is allowed to visualize the values of state variables and rewards during simulation and also alter the values of state variables. This animation facility is a powerful teaching and debugging tool even for analytical models. Figure 3(b) shows the simulation results of a RED model (different from that described in Figure 2) where the queue is modeled as a fluid, using the reward rate construct.

Besides conventional simulation, TANGRAM-II implements the RESTART (REpetitive Simulation Trials After Reaching Thresholds) technique [60] to improve simulation efficiency when statistics of rare events are of interest. The choice of RESTART was due to its simplicity and efficacy.

Solving computer network models using traditional simulation methods could be unfeasible due to the high computational cost to obtain the measures of interest when rates of different events in the model span orders of magnitude. This is the case, for instance, in packet simulation of high speed networks. To mitigate this problem, a technique called *fluid simulation* has been proposed [38] and can lead to a significant reduction in the computational effort.

In the fluid simulation technique, instead of representing the individual packets in the network, the traffic is viewed as a fluid that flows through a series of reservoirs. Therefore, events associated with the arrival and departure of packets to and from routers are not explicitly executed. Instead, only the resulting actions from changes in arrival and departure rates are executed and the equations that govern the changes in the fluid levels in each reservoir are solved. This can significantly reduce the time to simulate a model.

The modeling paradigm of the TANGRAM-II fluid simulator is based on reward rates [10]. To our knowledge, only a few tools based on fluid simulation have been developed. (See for instance, NETSIMUL [22], FluidSim [32] and references therein.), and both NETSIM, and FluidSim are tailored for networking modeling. TANGRAM-II, instead, implements object level constructs, that can be used to create objects tailored to different application domains.

The fluid simulator was built on top of the TANGRAM-II event driven simulator and inherited all the facilities and the power of the TANGRAM-II tool. It is generic enough to model complex systems and yet hides the complex fluid equations from the end user.

The paradigm used to build fluid models is based on the representation of the fluid reservoirs through reward rates. As such, rewards accumulate values which indicate the volume of fluid in a reservoir. Objects in TANGRAM communicate by exchanging messages. These can include any data, for instance, a new rate for the fluid emanating from A. As an example, suppose object A is sending a fluid to object B at a certain rate. Also suppose a given event occurs at time t in A that causes a change in the flow rate

to B. Then, a message is sent from object A to B at t indicating that, starting from t, B will receive the fluid from A at the new rate. Using the cumulative reward to represent a reservoir in B and messages to notify rate changes, the user has available the main building blocks needed to construct generic fluid simulation objects and build a hybrid fluid/conventional simulator. We know of no other tool that have similar flexibilities.

Some network fluid objects are available in the TANGRAM-II objects library, such as traffic sources, GPS and FIFO queues, communication links and a fluid leaky bucket. However, other objects can be easily built using the features implemented to support the simulation environment.

Several new constructs were developed to support the fluid simulator paradigm and to allow the implementation of new fluid objects. The most important of them are new forms to trigger an event, in particular when specified rewards reach given values.

From the fluid theory, it is necessary to know when a reservoir reaches a given threshold and triggers some action(s) to be executed when this event occurs. For example, the reservoir output fluid rate must be updated as soon as the reservoir empties. The event type called *reward_reached* occurs when the specified cumulative reward reaches a certain threshold, provided that conditions specified by the user are satisfied. The tool automatically computes the elapsed time until the occurrence of this event. This calculation is based on the values of the instantaneous ($IR(t)$) and cumulative rewards ($CR(t)$) and the specified limit (L).

The TANGRAM-II fluid simulator also makes constructs available to the user to control the sum of the specified cumulative rewards. This is useful to model complex queue management policies. For instance, to model a GPS queue with a finite buffer fed by two or more sources, the object *queue* must keep track of the total fluid in its reservoir from different sources, as well as the individual amounts of fluid stored to compute the input and output rate.

Figure 5 shows a model example which can be built using the TANGRAM-II fluid simulator. It is composed of a router fed by two On-Off sources which are policed by a leaky bucket. The queueing discipline is GPS. A sample of the simulation results obtained from this model includes the mean queue delay, the fraction of fluid lost and the router utilization. Trace files can also be automatically generated with instantaneous and cumulative reward values. Figure 6.(a) illustrates the instantaneous reward values of the leaky bucket object. They represent the arrival and departure rate of the leaky bucket, the bucket state and the input queue. Figure 6.(b) shows the cumulative rewards representing the

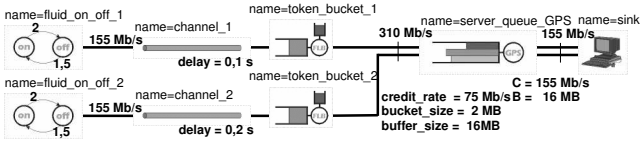

Figure 5: GPS queue fed by two policed On-Off sources

arrivals and departures of the leaky bucket as a function of time. We can observe the traffic shaping effect and the lower and upper bound values of the departure rate. These plots are very useful to evaluate the behavior of the fluids as a function of time.

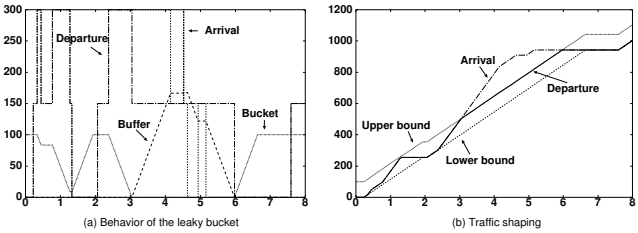

Figure 6: Traces obtained from the Leaky Bucket.

5. TRAFFIC ENGINEERING

The TANGRAM-II traffic engineering environment includes methods for calculating measures useful for traffic model as well as an extensive set of tools for experimentation and collecting parameters from the Internet. Its main features are described in what follows.

5.1 Traffic Modeling

Traffic characterization and modeling has been an extensive area of research and a lot of work has focused on obtaining accurate traffic models for dimensioning the network resources. Among these models, Markovian fluid-flow models (indeed rate reward models) have been widely and successfully employed. These are models for which a reward rate is associated with each state representing, for instance, the traffic flow at a given channel. One way of parameterizing the model is to match the statistics of the real traffic observed against those from the model. For these models $CR(t)$ is the total traffic flowing through the channel in $(0, t)$. If in turn we subtract from each reward a constant C equal to the transmission capacity of the channel, then $CR(t)$ is the channel queue size at t, provided that $CR(t)$ is bounded, i.e. $0 \leq CR(t) \leq B$ where B is the maximum buffer size.

A large set of TANGRAM-II modules is dedicated to traffic modeling and analysis. For instance, the user has the ability to create fluid-flow traffic models, obtain first and second order descriptors analytically or from a trace. Traffic descriptors such as the auto-covariance function $\mathrm{Cov}(t, \tau) = E[IR(t), IR(t+\tau)] - E[IR(t)]^2$ (for a stationary process, $\mathrm{Cov}(\tau) \lim_{t\to\infty} \mathrm{Cov}(t, \tau)$), and the index of dispersion for counts $\mathrm{IDC}(\tau) = \mathrm{Var}[N(\tau)]/E[N(\tau)]$ for a given time lag τ (where $IR(t)$ is the traffic rate (instantaneous reward) at t and $N(\tau)$ is the process that counts the amount of traffic transmitted during $(0, \tau)$). The tool uses the theory in [41] as the basis for the analytical calculations of the above measures.

The user can create a performance model using the traffic source under study and analyze the impact of the traffic on the resources during an observation period. Figure 7(a) shows an example where the auto-covariance is plotted for two Markovian traffic models and from one trace, as a function of the time lag. The right hand side of the figure shows the distribution of the buffer size calculated from the fluid-flow traffic model using the technique of [19].

5.2 Measurements and experimentation

The modeling process is not complete without carefully choosing the parameter values for the model under construc-

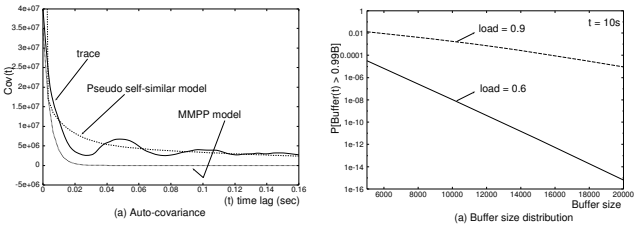

Figure 7: Example of application

tion. Obtaining such values may not be an easy task and many tools exist that support collecting statistics from computer systems. Without integration of such tools the modeling environment is not complete.

Due to the large number of specific problems a measurement study must face, measuring tools are most often targeted to the particular application domain of interest. For instance, tools for evaluating the performance of parallel systems, and tools for collecting measurements over the Internet. It is evident that there is a huge amount of work devoted to measurement techniques and tools. Here we focus on a single domain: active measurements over the Internet. This choice permits us to describe a few of the issues involved in collecting measurements useful to build a performance model, in an area that is rich in problems of current research interest.

During an *active measurement* statistics are collected through the use of *probes* (special packets) that are sent through the network from specified sources to data collectors located either at the source of the probe or at specified destinations. The advantage of this technique to collect statistics is that it usually does not rely on sophisticated data gathering equipment (*passive measurement* equipment) or *packet sniffers* that "watch" packets traversing a link. (Note that sniffers must operate at high rates, compatible with the speed of the link of interest.) Furthermore, the analyst may often not be able to place the equipment in the needed locations for collecting the measures of interest. Since an active measurement process introduces extra load in the network under observation, obviously care must be taken to avoid the extra load of the probe traffic interfering with the measurements.

A potential problem for obtaining certain results from active measurements is the inaccuracy of the measurements obtained due to, for example, unsynchronized clocks, cross traffic, etc. Therefore, sophisticated algorithms have been developed to cope with these problems, and a huge amount of literature exists on this subject area.

Several network statistics can be obtained from active probing. Measurements such as the round trip time (RTT) and the fraction of lost packets, are easy to calculate. However, measures such as the *one-way delay*, link speed estimation, etc., require special techniques to obtain the desired result.

In recent years, active measurement techniques and tools have been developed, as a result of many efforts to measure the Internet [52, 1, 6]. Some existing tools are simple to use and implement straightforward algorithms such as *traceroute*, *ping* and *bing*.

TANGRAM-II provides an environment to perform active measurements which includes a traffic generator and a module to calculate statistics. Some of the measurement techniques require that the probes be generated in a specific manner. For example, a recently proposed technique to estimate the bottleneck link capacity along a path requires that two consecutive probes are generated at regular intervals. Therefore, the analyst must choose the packet generation model according to the statistics she wants to collect.

In TANGRAM-II Traffic Generator packets can be generated from different source models such as: CBR (constant bit rate), Markovian, packet pair, or from a trace provided by the user. It is important to note that the packet generation module in TANGRAM-II is not only useful as a supporting facility to active probing but is also useful to provide an artificial load for other purposes. For instance, the user can generate traffic according to the same traffic model used in a performance model. This traffic could be send to an specific equipment such as a router, a switch or a WWW server. Examples of simple performance studies are: to generate traffic to determine buffer statistics of a given equipment, the behavior of the equipment when submitted to a variety of traffic conditions, etc.

Among the statistics that can be obtained with TANGRAM-II are: jitter, distribution of packet losses, RTT, one-way delay, bottleneck capacity of a path, and bottleneck buffer size. In what follows we discuss issues involved in collecting some of these statistics. It is interesting to note that, in order to estimate some of the metrics, probes may be generated in different directions between two points (one-way, two-way or round-trip), and employing different traffic generation patterns/models (CBR, Markovian, packet pair, or from a trace). The structure of the active measure traffic generation is shown in Figure 8.

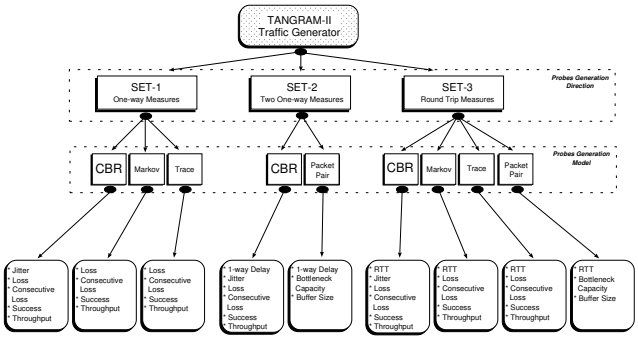

Figure 8: Tangram-II Traffic Generator Structure.

The computation of the one-way delay is based on time stamps collected at the source and at the destination of the probes. If the clocks of the transmitter and receiver are perfectly synchronized, then the one-way delay of a packet probe is simply the difference between two time stamps for a packet: one issued at the time the probe is transmitted and the other at the arrival time at the destination. However, unless special equipment is used both at the transmitter and receiver (such as GPS - Global Positioning System), the two clocks involved in the measurement are not synchronized. The difference in time between two unsynchronized clocks is called *clock offset* and the difference in the rate at which each clock advances is the *clock skew*.

Some techniques have been developed to cope with unsynchronized clocks. One of them is the Network Time Protocol (NTP). One problem with NTP is the accuracy of the syn-

chronization algorithm that depends on the frequency the clock information is exchanged between machines and on the delay between the hosts involved. Other approaches are based on techniques to estimate and remove the clock skew and offset. In the literature, several algorithms have been proposed to address the problems of offset estimation and skew correction [51, 59, 50, 61, 46].

In TANGRAM-II the techniques of [61] and [59] were combined and implemented to remove both the clock skew and offset and extended to handle additional issues mentioned below. Roughly [61] works as follows: packets are generated from a given source to a destination and the difference between the transmission and received time (measured delay) is collected for each. Let $\mathcal{S}$ be a set of (i, d_i) pairs, one for each probe sent, that indicates the packet sequence number and its measured delay. The lower boundary of the convex hull formed from the pairs in $\mathcal{S}$ is computed. The skew can be easily estimated from the points that belong to the lower boundary of this convex hull.

Not only the skew but also the offset must be removed from the measured delay. TANGRAM-II implements the technique of [59] since, to our knowledge, it is the only one in the literature that can handle asymmetric paths. The algorithm requires that the two hosts involved in the measurement exchange probes with different sizes and collect their delays.

Accurately estimating the one-way delay requires additional issues to be addressed, besides determining the skew and offset between the clocks. For instance, clock updates are not uncommon during the measurement period. Furthermore, received packets may not get time stamped immediately upon arrival. These problems cause inaccuracies in the process of calculating the one-way delay. All these issues are addressed in TANGRAM-II, and the above techniques were extended [56].

To illustrate the potential of the techniques and the tool one-way-delay (OWD) measures were collected between laboratories in Brazil and in the USA. Figure 9 shows the OWD measured between machines at the Federal University of Rio de Janeiro (UFRJ) and the University of Massachusetts at Amherst (UMass). The figure also plots different distributions for comparison.

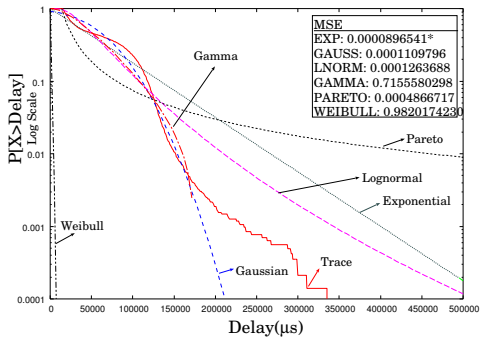

Figure 9: Distribution of the one-way delay from UMASS to UFRJ

The active measurement environment can aid the user in the process of parameterizing a set of *known* distributions from the measured OWD trace. The collected data can be used to obtain the distributions that best fit the measures. The method of moments is employed to estimate the parameters of six different distributions: Exponential, Gaussian, Lognormal, Pareto, Gamma and Weibull. Figure 9 plots several distributions obtained from a trace after the parametrization is performed. The Mean Squared Error (MSE) for each of the distributions is also available to the modeler. In the figure the MSE is indicated for each distribution.

Another useful measure that can be obtained with active probing is the capacity of the bottleneck link in a path. The technique called *packet pair* was introduced in [35] to estimate this measure. Tools such as *bprobe*, *Nettimer* and *sprobe* implement this technique. Briefly, packet pair requires the transmission of two consecutive packets (a packet pair) generated within a very short interval of time. The method calculates the capacity of a path based on the dispersion of the packets collected at the destination. The capacity of the bottleneck link is estimated from the difference between the arrival times of a packet pair and the size of the probes.

The so called *packet train* method is an extension of the packet pair approach and it is used by tools such as *cprobe*, *pathrate*, and *pipechar*. Both the packet pair and packet train techniques have been extensively used in the literature to estimate not only the bottleneck link capacity in a path but also measures such as the buffer size of the bottleneck router [5, 39, 21, 28].

These techniques are simple but can produce inaccurate results. The main reason is the packet pair jitter introduced by the cross traffic in the routers. A few works in the literature [21, 39, 28] have evaluated the precision of these methods and have shown that, under heavy load conditions, the cross traffic introduces errors on the capacity estimation with high probability.

Packet pair based techniques are implemented in the TANGRAM-II traffic engineering environment. To improve the accuracy of the original packet pair method, it was developed a packet selection algorithm to be used with the basic packet pair method [57]. Roughly, the tool employs OWD estimates to selected the packet pairs that will be used in the calculations. That is, a pair is selected for the packet pair estimation procedure when the estimated OWD of its first packet is within a given tolerance from the smallest OWD of the entire measurement samples. This was shown to produce accurate results.

6. CONCLUSIONS

Developing tools for performance/dependability analysis of computer systems and networks has always been a challenging task. The modeling cycle encompasses several issues from creating an abstract representation of the system under study, parameterizing it, perhaps using measures collected in the real system, solving the model efficiently, changing the model parameters and including additional modeling detail, running experiments, analyzing the data collected, etc.

In the development of the TANGRAM-II tool we tried to create an environment that addresses a wide range of issues in the modeling process and yet is easy to use. These include issues related to model specification, the set of analytic and simulation solution techniques made available to the modeler, and finally performing measurements.

TANGRAM-II was built to provide the user with a wide variety of building blocks to assist in the complete life cycle

of the modeling process: build simple analytic models to understand a problem, experiment with different solution techniques, parametrize the model by collecting measurements, build more complex simulation models, and gain confidence in the model results by comparing simple analytic and complex simulation models.

Briefly, our contributions include: (a) the idealization of an integrated, easy to use generic environment for handling from model specification to solutions and experimentation and; (b) the development of new algorithms. These include transient and steady state solution algorithms, a fluid simulator coupled with the event driven simulator for handling generic models, and new techniques for active measurements. Furthermore, the tool has a rich set of functionalities for aiding in the calculation and visualization of a variety of measures of interest.

Augmenting the basic modeling environment of TANGRAM-II, network application tools have been developed such as a voice/video transmission tool and a distributed whiteboard. These support experimentation, collaborative modeling, and data collection in a real network environment. Furthermore, new algorithms can be easily added as new techniques are developed, and new objects can be included without much effort as part of existing or new application domains.

7. REFERENCES

[1] A. Adams, T. Bu, T. Friedman, J. Horowitz, D. Towsley, R. Cáceres, N. Duffield, and F. Presti. The Use of End-to-end Multicast Measurements for Characterizing Internal Network Behavior. *IEEE Communications Magazine*, 38(5), 2000.

[2] D. Anick, D. Mitra, and M. M. Sondhi. Stochastic theory of a data-handling system with multiple sources. *The Bell System Technical Journal*, 61(8):1871–1894, 1982.

[3] S. Berson, E. de Souza e Silva, and R. Muntz. An object oriented methodology for the specification of Markov models. In *Numerical Solution of Markov Chains*, pages 11–36. Marcel Dekker, Inc., 1991.

[4] R. M. L. R. Carmo, E. de Souza e Silva, and R. Marie. Efficient solutions for an approximation technique for the transient analysis of Markovian models. Technical report, IRISA, no. 3055, Campus universitaire de Beaulieu, 1996.

[5] R. L. Carter and M. E. Crovella. Measuring bottleneck link speed in packet-switched networks. In *Performance Evaluation*, 1996.

[6] W. Chen, Y. Huang, B. Ribeiro, K. Suh, H. Zhang, E. de Souza e Silva, J. Kurose, and D. Towsley. Exploiting the IPID Field to Infer Network Path and End-System Characteristics. *Lecture Notes in Computer Science*, 3431:108–120, 2005.

[7] G. Chiola, G. Franceschinis, R. Gaeta, and M. Ribaudo. GreatSPN 1.7: Grafical editor and analyzer for timed and stochastic Petri nets. *Performance Evaluation*, 24(1-2):47–68, 1995.

[8] G. Ciardo, J. Muppala, and K. Trivedi. The stochastic Petri net package, version 3.1. In *Proceedings of MASCOT'93*, pages 390–391, 1993.

[9] P. Courtois and P. Semal. Computable bounds for conditional steady-state probabilities in large Markov chains and queueing models. *IEEE JSAC*, 4(6):926–937, 1986.

[10] K. de Freitas Reinhardt, E. de Souza e Silva, and R. M. M. Leão. A Fluid Simulation Environment for Multimedia Networks. In *XXI Brazilian Symposium on Communication Networks (in portuguese)*, pages 119–134, Brazil, Maio 2003.

[11] E. de Souza e Silva and H. Gail. Analyzing scheduled maintenance policies for repairable computer systems. *IEEE Trans. on Computers*, 39(11):1309–1324, 1990.

[12] E. de Souza e Silva and H. Gail. Performability analysis of computer systems: from model specification to solution. *Performance Evaluation*, 14:157–196, 1992.

[13] E. de Souza e Silva and H. Gail. An algorithm to calculate transient distributions of cumulative rate and impulse based reward. *Stochastic Models*, 14(3):509–536, 1998.

[14] E. de Souza e Silva and H. Gail. Transient Solutions for Markov Chains. In W. Grassmann, editor, *Computational Probability*, pages 44–79. Kluwer, 2000.

[15] E. de Souza e Silva, H. Gail, and R. Muntz. Efficient solutions for a class of non-Markovian models. In W. Stewart, editor, *Computations with Markov Chains*, pages 483–506. Kluwer, 1995.

[16] E. de Souza e Silva, H. Gail, and R. Muntz. Polling systems with server timeouts and their application to token passing networks. *IEEE Trans. on Networking*, 3(5):560–575, 1995.

[17] E. de Souza e Silva, H. Gail, and R. Muntz. Gated time-limited polling systems. In T. Hasegawa, H. Takagi, and Y. Takahashi, editors, *Performance and Management of Complex Communication Networks*, pages 253–274. Chapman & Hall, 1998.

[18] E. de Souza e Silva and H. R. Gail. The Uniformization Method in Performability Analysis. In G. R. B. R. Haverkort, R. Marie and K. S. Trivedi, editors, *Performability Modelling: Techniques and Tools*, chapter 3, pages 31–58. Wiley, 2001.

[19] E. de Souza e Silva, R. M. M. Leão, and M. C. Diniz. Transient Analysis Applied to Traffic Modeling. *Performance Evaluation Review*, 28(4):14–16, 2001.

[20] D. D. Deavours, G. Clark, T. Courtney, D. Daly, S. Derisavi, J. M. Doyle, W. H. Sanders, and P. G. Webster. The möbius framework and its implementation. *IEEE Trans. on Software Engineering*, 28(10):956–969, 2002.

[21] C. Dovrolis, P. Ramanathan, and D. Moore. What do packet dispersion techniques measures? In *IEEE Infocom 2001*.

[22] D. R. Figueiredo, B. Liu, Y. Guo, J. Kurose, and D. Towsley. On the efficiency of fluid simulation of networks. *Computer Networks*, 50(12):1974–1994, August 2006.

[23] S. Floyd and V. Jacobson. Random early detection gateways for congestion avoidance. *IEEE/ACM Trans. Netw.*, 1(4):397–413, 1993.

[24] G. Franceschinis and R. Muntz. Bounds for Quasi-Lumpable Markov Chains. In *Proc. Performance '93*, Rome, Italy, Sept 1993.

[25] W. Grassmann. The GI/PH/1 queue: a method to find the transition matrix. *INFOR*, 20(2):144–156,

1982.

[26] W. Grassmann. Finding transient solutions in Markovian event systems through randomization. In W. J. Stewart, editor, *Numerical Solution of Markov Chains*, pages 357–371. Marcel Dekker, Inc., 1991.

[27] W. Grassmann, M. Taksar, and D. Heyman. Regenerative analysis and steady state distributions for Markov chains. *Operations Research*, 33(5):1107–1116, 85.

[28] K. Harfoush, A. Bestravos, and J. Byers. Measuring bottleneck bandwidth of targeted path segments. In *IEEE Infocom 2003*.

[29] B. Haverkort. Performability evaluation of fault-tolerant computer systems using DyQNtool+. *Int'l Journal Reliability, Quality and Safety Eng.*, 2(4):383–404, 1995.

[30] P. Heidelberger. Fast simulation of rare events in queueing and reliability models. *ACM Transactions on Modeling and Computer Simulation*, 5(1):43–85, 1995.

[31] P. Heidelberger and A. Goyal. Sensitivity analysis of continuous time Markov chains using uniformization. In G. Iazeolla, P. Courtois, and O. Boxma, editors, *Computer Performance and Reliability*, pages 93–104. North-Holland, 1988.

[32] J. Incera, R. Marie, D. Ross, and G. Rubino. FluidSim: A Tool to Simulate Fluid Models of High Speed Networks. *Performance Evaluation*, 44:25–49, 2001.

[33] Insecure.org. Network Simulator. http://www.isi.edu/nsnam/ns.

[34] K. Irani and V.L.Wallace. On network linguistics and the conversational design of queueing networks. *Journal of the ACM*, 18:616–629, 1971.

[35] V. Jacobson. Congestion avoidance and control. In *ACM SIGCOMM 1988*, pages 314–329.

[36] J.C.S. Lui and R.R. Muntz. Bounding Methodology for Computing Steady State Availability of Repairable Computer Systems. *Journal of the ACM*, 41(4):676–707, 1994.

[37] R. L. Klevans and W. J. Stewart. From queueing networks to markov chains: The XMARCA interface. *Performance Evaluation*, 24(1-2):23–45, 1995.

[38] K. Kumaram and D. Mitra. Performance and Fluid Simulations of a Novel Shared Buffer Management System. In *IEEE Infocom 1998*.

[39] K. Lai and M. Baker. Measuring bandwidth. In *IEEE Infocom 1999*.

[40] G. Latouche. Algorithms for Infinite Markov chains with Repeating Columns. In *Linear Algebra, Markov Chains, and Queueing Models*, pages 231–266. 1993.

[41] R. M. M. Leão, E. de Souza e Silva, and S. C. de Lucena. A Set of Tools for Traffic Modeling, Analysis and Experimentation. In *Lecture Notes in Computer Science*, volume 1786, pages 40–55. Springer, March 2000.

[42] C. Lindemann. DSPNexpress: A software package for the efficient solution of deterministic and stochastic petri nets. *Performance Evaluation*, 22(1):3–21, 1995.

[43] M. Martinello and E. de Souza e Silva. A testbed for network performance evaluation and its application to connection admission control algorithms. *Journal of the Brazilian Computer Society*, 7(2):39–51, 2001.

[44] C. Meyer. Stochastic complementation, uncoupling Markov chains, and the theory of nearly reducible systems. *SIAM Review*, 31(2):240–272, 1989.

[45] J. Meyer. On evaluating the performability of degradable computing systems. *IEEE Trans. on Computers*, C-29(8):720–731, 1980.

[46] S. B. Moon, P. Skelly, and D. Towsley. Estimation and removal of clock skew for network delay measurements. In *IEEE Infocom 1999*.

[47] R. Muntz, E. de Souza e Silva, and A. Goyal. Bounding availability of repairable computer systems. *IEEE Trans. on Computers*, 38(12):1714–1723, 1989.

[48] M. F. Neuts. *Matrix-geometric Solutions in Stochastic Models – an Algorithmic Approach*. John Hopkins University Press, Baltimore, MD, 1981.

[49] P. Buchholz and G. Ciardo and S. Donatelli and P. Kemper. Kronecker Operations and Sparse Matrices with Applications to the Solution of Markov Models. *INFORMS Journal on Computing*, 12(3):203–222, 2000.

[50] A. Pasztor and D. Veitch. PC Based Precision Timing Without GPS. In *ACM Sigmetrics 2002*.

[51] V. Paxson. On calibrating measurements of packet transit times. In *ACM Sigmetrics 1998*.

[52] V. Paxson, A. Adams, and M. Mathis. Experiences with NIMI. In *Proc. of Passive and Active Measurement*, 2000.

[53] B. Plateau. On the stochastic structure of parallelism and synchronization models for distributed algorithms. In *ACM Sigmetrics 1985*.

[54] A. Riska and E. Smirni. MAMSolver: a Matrix Analytic Methods Tool. In *Lecture Notes in Computer Science*, volume 2324, pages 205–211. Springer, 2002.

[55] A. Riska and E. Smirni. M/G/1-Type Markov Processes: A Tutorial. In *Performance Evaluation of Complex Systems: Techniques and Tools*, volume 2459 of *LNCS*, pages 36–63. 2002.

[56] A. A. A. Rocha, R. M. M. Leão, and E. de Souza e Silva. A Methodology to Estimate One-way Delay and Internet Experiments (in portuguese). In *XXII Brazilian Computer Network Symposium(SBRC'04)*, Gramado, Brazil, May 2004.

[57] A. A. A. Rocha, R. M. M. Leão, and E. de Souza e Silva. A New Technique to Select Packet Pairs to Estimate Bottleneck Link Capacity (in portuguese). In *III WPerformance/XXIV SBC*, Salvador, Brazil, August 2004.

[58] W. Stewart. *Introduction to the Numerical Solution of Markov Chains*. Princeton University Press, 1994.

[59] M. Tsuru, T. Takine, and Y. Oie. Estimation of clock offset from one-way delay measurement on asymmetric paths. In *SAINT International Symposium on Applications and the Internet*, 2002.

[60] M. Villén-Altamirano and J. Villén-Altamirano. RESTART: A Straightforward Method for Fast Simulation of Rare Events. In *Proceedings of the 1994 Winter Simulation Conference*, pages 282–289.

[61] L. Zhang, Z. Liu, and C. H. Xia. Clock synchronization algorithms for network measurements. In *IEEE Infocom 2002*.