PERFORMANCE ANALYSIS OF THE KNOCKOUT SWITCH UNDER BURSTY TRAFFIC BASED ON A STOCHASTIC ACTIVITY NETWORK MODEL *

Latha Kant† and William H. Sanders‡
†Bellcore
445 South Street
Morristown, NJ 07960
lkant@bellcore.com, Tel: (973) 829-2879
‡ Center for Reliable and High-Performance Computing
Coordinated Science Laboratory
University of Illinois
Urbana, IL 61801
whs@crhc.uiuc.edu, Tel: (217) 333-0345

ABSTRACT

The possibility of ATM-based B-ISDNs has made desirable the integration of diverse services like voice, data, and multi-media. The success of these high-speed networks is critically dependent on the availability of fast packet switches (FPS), with the capacity to switch and route at high speeds *and* provide the required quality of service (QoS). The diversity of the applications, however, leads to widely differing QoS demands. The knockout switch is an FPS with low cell-loss probability and latency under steady input traffic. However, in light of the bursty nature of traffic in B-ISDNs, it is important to analyze the switch performance under varying intensities and mixes of correlated inputs. In this paper, we analyze the performance of the knockout switch in terms of cell-loss probabilities and distribution of queue length for a wide range of burst parameters. The switch is modeled using stochastic activity networks (SANs), and the underlying Markov processes are generated and solved using *UltraSAN*, a SAN-based performance and dependability evaluation software package.

*This work was supported, in part, by a grant from Bell Communications Research.

The results provide useful information about the switch sensitivity and robustness under a variety of burst parameters, and illustrate the appropriateness of SANs for modeling and analyzing telecommunication switch architectures.

Keywords: asynchronous transfer mode, broadband ISDN, cell-loss probability, distribution of queue length, fast packet switch, stochastic activity networks, stochastic Petri nets.

I Introduction

Integration of diverse applications such as voice, multi-media, and data on a single transport network is being made possible by the proposed high-speed ATM-based B-ISDNs. However, these diverse applications have widely differing quality of service (QoS) requirements. Real-time applications like voice and video typically require low delays but are more resilient to loss, while the situation is the reverse with data. This places stringent performance requirements on the associated ATM switches, and has lead to extensive research in the design of fast packet switches (FPS).

A variety of FPSs have been proposed [1, 15, 20], and, as pointed out by [15], a unique classification of the FPS designs is difficult to obtain. Further, the choice of a particular FPS is not simple either, due to tradeoffs in cost and performance. Specifically, while switches with input queueing are fairly inexpensive to implement [2], they suffer from HOL blocking [7]. Output queueing, on the other hand, does not exhibit HOL blocking.

In light of the high bandwidths in B-ISDNs, significant emphasis is laid on switches with good throughput (low blocking) characteristics. We therefore select the knockout switch proposed in [22] for our analysis. The knockout switch is an FPS with low cell loss and latency under steady input traffic. Other features of this switch include a fully interconnected architecture (avoiding internal blocking), output queueing (avoiding HOL blocking), synchronous input/output ports (avoiding any internal speed-ups) and, a knockout mechanism, along with the ability to be easily fabricated and configured into large modules. The first two features imply good throughput and the last ensures viable cost. (See Section II for a description of the switch.)

Prior work on the knockout switch includes: (a) analysis with variable length packets and uniform routing [3], (b) simulation and implementation of a prioritized knockout switch [4], and (c) analysis with non-uniform routing and fixed length packets [23]. Studies (a)

and (c) consider i.i.d. Bernoulli inputs, which are not suitable in B-ISDNs [21]. While the work by Evans *et al* [4] did consider bursty traffic, since simulation was used, it was limited to small concentrator sizes and high error rates to avoid extremely long run times. Further, the focus was on the implementation of a prioritized switch and it did not analyze switch performance and sensitivity to a broad range of burst parameters, as is typical in integrated services networks.

Since the specific mix (hence burstiness) at the FPS is hard to predict, it is very important that switch analyses account for both traffic correlations as well as variations in burstiness. In this paper, we analyze the behavior of the knockout switch over a wide range of burst parameters to provide a more complete evaluation of the switch with correlated traffic. Further, we use the cell-loss probability (CLP) and the output queue length distribution (rather than the commonly used measure – the average queue length) as our performance measures. Our results are significant in that they provide useful information about switch sensitivity and robustness under a variety of burst parameters. Among other information, our results indicate that the sensitivities of the concentrator and output queue with varying burst intensities differ significantly from their steady traffic counterparts. Our analysis also confirms that though the average queue length is a sufficient indicator of performance (QoS) with steady traffic, it it not always appropriate with bursty traffic. Instead the distribution of the number of cells in the output queue is a better measure.

To help in the detailed characterization and analysis of the switch, efficient modeling techniques that can capture the architectural details without sacrificing accuracy is needed. We use stochastic activity networks (SANs) [12, 18], a variant of stochastic Petri nets, to model the knockout switch and a bursty workload. The SAN formalism allows for compact representation of switch architectures and workloads, and enables automatic generation of the associated stochastic processes. Since the size of these processes are in the order of

tens of thousands of states, an accurate hand construction becomes a formidable task. We use *UltraSAN* [18], a SAN-based performance modeling and analysis tool, to automate the generation of the underlying Markov processes and solve them numerically. Since these processes are solved easily on a typical workstation, we are able to construct and solve the detailed stochastic process representation of the FPS and workload accurately. Our results, therefore, not only shed useful insights to switch performance under bursty traffic, but also illustrate the powerful framework provided by SANs for evaluating FPS designs.

The remainder of the paper is structured as follows. Section II provides a brief overview of the knockout switch, followed by a description of the switch and workload model. The details of the SAN model developed are described in Section III. The model solution and results are discussed in Section IV, and Section V presents the concluding remarks.

II Switch and Workload Models

The knockout switch (see Figure 1) is an FPS with the following features [22]:

- A fully interconnected bus structure to avoid internal blocking.
- Separate queues with packet filters at each output port to filter cells not destined to it.
- A knockout concentrator at each output port with N input and L output lines, with $L \ll N$, typically, for an N input switch. A mechanism similar to a knockout tournament to discard excess over L cells destined to a particular port during a single cell time is employed. The L winning cells are placed in the output queue via the shifter mechanism, as proposed in the Knockout switch design [22].
- The cells at each output port are served FIFO.
- Easy modular growth and expandability to large switch sizes from a small basic unit.

A Switch Model

To model the knockout switch, we begin by looking at the precise sequence of events that takes place upon service completion. Before enumerating this sequence, our assumptions (which are reasonable for the FPS) are:

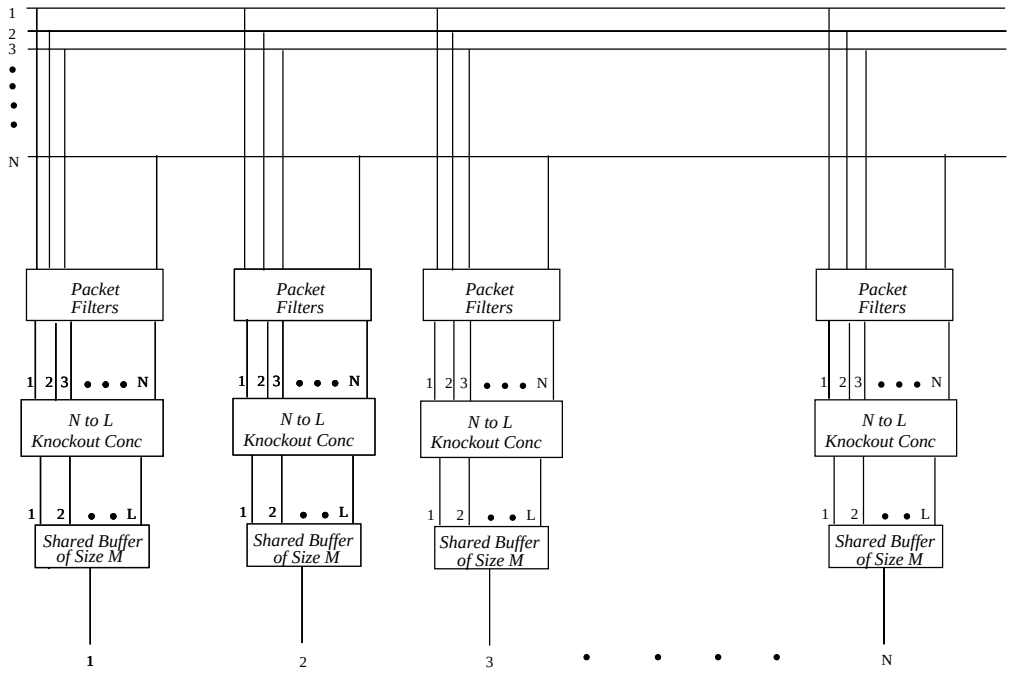

Figure 1: Knockout switch

1. Non pre-emptive transmission of cells.
2. Fixed and deterministic times at which discarding decisions are made, i.e., discarding decisions are performed immediately after every service completion, with the time for a service completion, i.e., cell transmission time, corresponding to one time slot.

Let n represent a time at which a discarding decision is made, and N the number of input and output ports. Since the ports operate synchronously, the sequence of events at each output port is:

1. The *(n-1)th* service completion (SC) occurs. The beginning of the nth SC is marked by the removal of the cell that was in service during the *(n-1)th* service interval (SI).
2. High-speed discarding (knockout) decision performed in negligible time.
3. Newly admitted cells join the output queue.
4. The *nth* SI begins, representing the service time for the cell at the head of the output queue.

Since the knockout switch employs separate output queues, we may restrict our attention to a single output port. Let the state of the system in the *nth* service interval be $X_n =$

(T_n, D_n, M_n), where:

1. T_n is the number of cells destined to the given output port at the beginning of the *nth* SI.

2. D_n is the number of cells discarded at the beginning of the *nth* SI. It is a function of cell admission policy at the switch (i.e.,the knockout algorithm with concentrator L), the maximum output buffer size (M_{max}), the number of cells queued at this output queue (M_{n-1}), and the number of cell arrivals T_n. Thus,

$$\begin{aligned} D_n &= 0 && T_n \leq L \leq M_{max} - M_{n-1} \\ &= T_n - L && T_n > L \quad and \quad L \leq M_{max} - M_{n-1} \\ &= T_n - (M_{max} - M_{n-1}) && otherwise \end{aligned}$$

3. M_n is the number of cells remaining in the output buffer after the *(n-1)th* SC, plus the cells that were just admitted to this output buffer, i.e.,

$$M_n = M_{n-1} + T_n - D_n - 1.$$

Since the system is viewed at discrete time intervals (corresponding to deterministic service completion times), and the next state X_{n+1} depends only upon the current state X_n, the process $\{ X_n \mid n \in I\!N \}$ is a discrete-time Markov chain (DTMC).

Defining the cell-loss probability at time n (CLP_n) as the fraction of the number of arriving cells discarded, we have

$$CLP_n = E[D_n]/E[T_n] \tag{1}$$

where $E[\cdot]$ denotes expectation. These expectations are obtained by solving for the state occupancy probabilities $\pi^{(n)}_{i,j,k}$, where $\pi^{(n)}_{i,j,k}$ denotes the probability that the system is in state $X_n = (i, j, k)$ at time n. The denominator in (1) is the traffic intensity ρ. Since the DTMC under consideration is finite state, irreducible, aperiodic, and time homogeneous, the limiting probabilities exist and are independent of the initial state. Thus, $\pi_{i,j,k} = \lim_{n\to\infty} \pi^{(n)}_{i,j,k}$, and (1) becomes

$$CLP = \frac{\sum_{i,j,k}\ j\ \pi_{i,j,k}}{\rho}. \tag{2}$$

Equation (2) gives the limiting CLP as seen by the switch.

B Workload model

Many bursty traffic models exist in the literature, ranging from among others [5, 6, 9, 11, 16, 17] characterized by Markovian models, to the more recent study in self-similar traffic [10] and long-range models [14]. The choice of an "appropriate" traffic model to capture the diverse traffic in B-ISDNs is however a matter of debate, as is the characterization of the input traffic mix to the FPS. Since we are interested in analyzing switch sensitivity to correlated traffic mix in general and not a particular application type, we employ a two-state Markov modulated Bernoulli process (MMBP) as in [9, 11]. In doing so, we vary the burst parameters of the workload over a wide range to capture the effect of multiplexing diverse applications.

The workload switches between idle and active states, representing spurts of idle and active periods of input traffic to a port. While active, a cell is emitted with probability q with a maximum of one cell being emitted during a time slot. When idle, no cell is emitted. The holding time in each state is geometrically distributed. The probabilities of remaining in or transitioning from the idle state are given by p_{00} and p_{01}, respectively. Likewise, p_{11} and p_{10}, respectively, denote the probability of remaining in or transitioning from the active state.

Let $\underline{\pi}$ denote the steady-state probability distribution vector. Thus,

$$\pi_0 = \frac{p_{10}}{p_{01} + p_{10}} \quad and \quad \pi_1 = \frac{p_{01}}{p_{01} + p_{10}}, \tag{3}$$

and the average time spent in an active period E_{active} (ON time) in cell times is

$$E_{active} = \frac{1}{p_{10}}. \tag{4}$$

The average offered load ρ is

$$\rho = \frac{p_{01}}{p_{01} + p_{10}} \times q. \tag{5}$$

with q denoting the cell arrival probability when active.

To characterize bursty traffic, we use the average burst size (BS) and activity factor (AF) as in [9, 11]. Defining the AF as the fraction of time the workload is active, we have

$$AF = \frac{E_{active}}{E_{active} + E_{idle}} \quad = \quad \pi_1, \tag{6}$$

and the average burst size is

$$BS = E_{active} \times q \quad = \quad \frac{1}{p_{10}} \times q. \tag{7}$$

The effect of bursty traffic is studied by varying the burst size (BS), activity factor (AF), and cell emission probability (q). Thus equations (4) through (7) help in selecting the appropriate Markov chain parameters while analyzing switch performance with bursty inputs.

III SAN Representation

The stochastic process representations of the switch and workload models described in Section II are on the order of tens of thousands of states. Hence, rather than build these detailed Markov processes by hand, we construct higher level SAN models and use *UltraSAN* to automate the generation and solution of the underlying Markov process. Figure 2 presents a stochastic activity network (SAN) representation for the knockout switch with bursty traffic. Since space does not permit us to describe the SAN formalism in detail, we briefly introduce the SAN components in the following two paragraphs. For further information on SANs, see [12, 13].

SANs consist of four primitives: *places, activities, input gates*, and *output gates*. Places, represented by circles e.g., *active* in Figure 2), represent the "state" of the modeled system and may contain *tokens*. Activities ("transitions" in Petri net terminology), denoted by vertical bars, represent actions of the modeled system and may be of two types: (a) *timed*

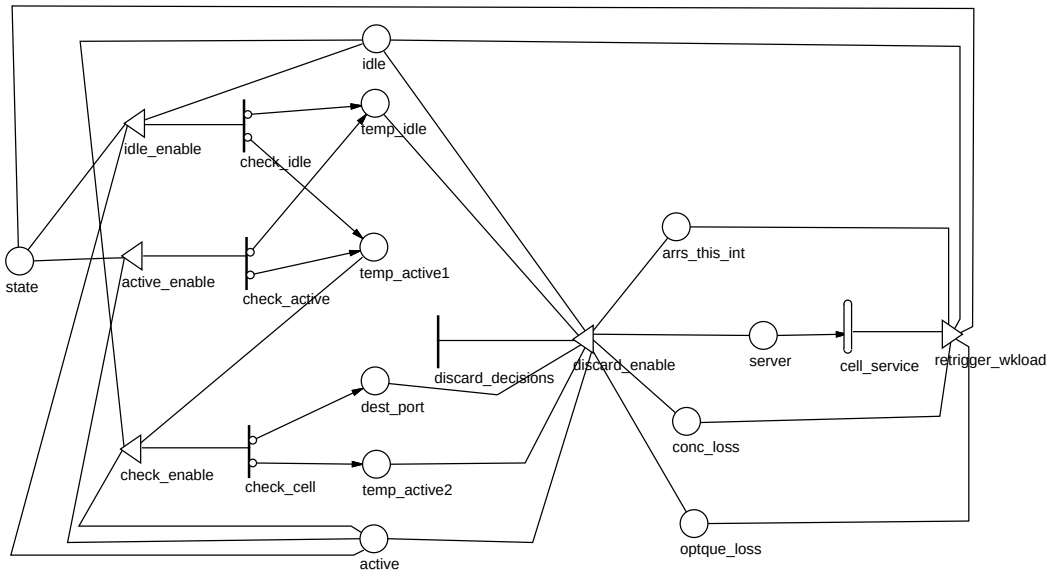

Figure 2: SAN model for the switch with bursty workload

and (b) *instantaneous*. Timed activities (denoted by hollow vertical bars) have durations which impact the performance of the modeled system, for example, the transmission time of an ATM cell, represented by *cell_service* in Figure 2. Instantaneous activities (denoted by a thick vertical bar) represent actions that complete in a negligible amount of time compared to other activities in a system. For example, *discard_decisions* in Figure 2 is an instantaneous activity representing high-speed hardware switching and routing within the FPS.

Case probabilities, represented as small circles on the right side of an activity, represent uncertainty associated with the completion of an activity, with each case representing a possible outcome. In Figure 2, *check_active* is an example of an instantaneous activity with two cases. *Input gates* are used to enable activities, while both input and output gates help change the state of the system. An example of an input gate is *active_enable* in Figure 2, which enables activity *check_active*, while *retrigger_wkload* is an example of an output gate.

The working of the SAN model is as follows. Places *idle* and *active* represent the idle

Gate	*Definition*
idle_enable	Predicate *MARK(state) == 1*
	Function *MARK(idle) − −;* *if(MARK(idle) == 0) {* *if(MARK(active) > 0)* *MARK(state) = 2;* *else* *MARK(state) = 0; }*

Table 1: Specification for input gate *idle_enable*

Gate	*Definition*
active_enable	Predicate *MARK(state) == 2*
	Function *MARK(active) − −;* *if(MARK(active) == 0)* *MARK(state) = 0;*

Table 2: Specification for input gate *active_enable*

and active states of the input, with the number of tokens in them totaling N. At the beginning of every SI, the number of tokens in place *state* is checked. If it contains 1 token (implying at least 1 source was idle in the *(n-1)th* SI), activity *check_idle* is enabled, otherwise *check_active* is enabled. In the former case, after the idle inputs are examined, the model checks to see if there are any remaining inputs that were active, in which case it then enables *check_active*. Tables 1 and 2 give the enabling predicate and functions for the input gates *idle_enable* and *active_enable*, respectively.[1]

The activities *check_idle* and *check_active*, along with their case probabilities, implement

[1]In tables 1 through 6, $MARK$, $GLOBAL\_S$ and $GLOBAL\_D$ are SAN key words. $MARK$ followed by the name of a place in parenthesis denotes the marking (number of tokens) in that place. $GLOBAL\_S$ and $GLOBAL\_D$ are used to declare global variables of type short, and double, respectively. Global variables in SANs allow for parameterization of the model.

Gate	*Definition*
check_enable	Predicate $(MARK(idle) == 0)\ \&\&(MARK(active) == 0)\&\&(MAR\ K(temp\_active1)) > 0$
	Function $MARK(temp\_active1)\ --;$

Table 3: Specification for input gate *check_enable*

Activity	*Case*	*Probability*
check_idle	1	$return(1.0 - GLOBAL\_D(IDLEtoACTIVE));$
	2	$return(GLOBAL\_D(IDLEtoACTIVE));$
check_active	1	$return(GLOBAL\_D(ACTIVEtoIDLE));$
	2	$return(1.0 - GLOBAL\_D(ACTIVEtoIDLE));$
check_cell	1	$return(GLOBAL\_D(CELLON));$
	2	$return(1.0 - GLOBAL\_D(CELLON));$

Table 4: Activity case probabilities

the four transition probabilities in the two-state burst model, i.e., cases 1 and 2 of *check_idle* represent p_{00} and p_{01}, respectively. Likewise, cases 1 and 2 of *check_active*, denote p_{10} and p_{11}, respectively. If an input becomes active, a token is placed in *temp_active1* and *check_cell* is enabled, which checks for the presence or absence of a cell. A cell is emitted with a probability q (case 1) and not with probability *(1-q)* (case 2) of *check_cell*. Places *dest_port* and *temp_active2* are incremented with the choice of cases 1 and 2, respectively. Table 3 gives the enabling predicate and function for input gate *check_enable* while Table 4 lists the transition probabilities for the workload as well as that for cell arrivals when the workload is active. The transition probabilities in Table 4 have been parameterized via $GLOBAL\_D()$, to vary the AFs, BS and q.

More specifically, the coding of the different case probabilities in Table 4 is as follows. Recall that the degree of workload burstiness may be selected via equations (4) through (7). Thus for a particular desired ON time, the value of p_{10} is obtained via equation (4). This

value is coded in case 1 for activity *check_active*. Case 2 of *check_active* is easily obtained as $(1.0 - Case1)$. Next, for a particular AF and ON time, the value of p_{01} is obtained from equation (6). This value of p_{01} is coded as the value for case 2 of activity *check_idle*, the value of case 1 for *check_idle* is obtained as $(1.0 - Case2)$. Finally, case 1 of activity *check_cell* is given the value of q while case 2 of activity *check_cell* is obtained as $(1.0 - q)$.

Once all the input ports have been examined, *discard_enable* enables *discard_decisions* to implement the discarding decisions at the concentrator (knockout mechanism) and at the output queue. Since routing and discarding are performed at high speeds, an instantaneous rather than timed activity is employed. The function of gate *discard_enable* performs the knockout algorithm as follows. It checks the number of cell arrivals over a given SI (obtained from *arrs_this_int*). If it exceeds L, it knocks out the excess over L cells, updates *conc_loss* (which represents the cells knocked out at the concentrator stage), and checks if the L cells can all be accommodated at the output queue. If yes, the cells join the output queue *server*, else the excess over M cells are discarded, and *optque_loss* (representing the cells lost at the output queue) is updated. If the number of cell arrivals is less than L, it sets *conc_loss* to zero, performs output queue fill check, and updates *optque_loss* if need be. The algorithm used (the enabling predicate and function of the input gate) is given in Table 5. Since we are interested in studying switch behavior with different concentrator and buffer sizes, their values have been parameterized via $LMAX$ and $QMAX$, respectively.

Timed activity *cell_service* represents the fixed time required to serve an ATM cell. Upon service completion, output gate *retrigger_wkload* is enabled, which resets *conc_loss,optque_loss*, and *arrs_this_int*. It then checks to see if any sources are idle, and if yes, places one token in *state*, else two tokens, triggering the workload for the next SI. These functions are given in Table 6.

Gate	*Definition*
discard_enable	Predicate $(MARK(dest\_port)+MARK(temp\_idle) + MARK(temp\_active2) == GLOBAL\_S(INPTS))$

	Function
	$MARK(arrs\_this\_int) = MARK(dest\_port);$
	$MARK(idle) = MARK(temp\_idle);$
	$MARK(active) = MARK(temp\_active2) + MARK(dest\_port);$
	$if(MARK(arrs\_this\_int) > GLOBAL\_S(LMAX))$ {
	$MARK(conc\_loss) = MARK(arrs\_this\_int) - GLOBAL\_S(LMAX);$
	$if(GLOBAL\_S(QMAX)-MARK(server) >= GLOBAL\_S(LMAX))$ {
	$MARK(server) += GLOBAL\_S(LMAX);$
	$MARK(optque\_loss) = 0;$ }
	$else$ {
	$MARK(optque\_loss) = GLOBAL\_S(LMAX)-(GLOBAL\_S(Q\ MAX)-MARK(server));$
	$MARK(server) = GLOBAL\_S(QMAX);$ } }
	$else$ {
	$MARK(conc\_loss) = 0;$
	$if((GLOBAL\_S(QMAX)-MARK(server)) >= MARK(arrs\_this\_int))$ {
	$MARK(server) += MARK(arrs\_this\_int);$
	$MARK(optque\_loss) = 0;$ }
	$else$ {
	$MARK(optque\_loss) = MARK(arrs\_this\_int) - (GLOBAL\_S(QMAX)-MARK(server));$
	$MARK(server) = GLOBAL\_S(QMAX);$ } }
	$MARK(dest\_port) = 0;$
	$MARK(temp\_active2) = 0;$
	$MARK(temp\_idle) = 0;$
	$if(MARK(server) == 0)\ MARK(server)++;$

Table 5: Specification for input gate *discard_enable*

Gate	*Definition*
retrigger_wkload	$if(MARK(idle) > 0)$ $MARK(state) = 1;$ $else$ $MARK(state) = 2;$ $MARK(arrs\_this\_int) = 0;$ $MARK(optque\_loss) = 0;$ $MARK(conc\_loss) = 0;$

Table 6: Specification for output gate *retrigger_wkload*

IV Model Solution and Discussion of Results

The SAN models presented in the preceding section are analyzed by constructing and solving the appropriate Markov chains. This is performed by *UltraSAN* [18], which automates the construction of the corresponding continuous time Markov process (CTMP). As mentioned in the introduction, *UltraSAN* is a software package for model-based evaluation of systems (in this case, for the knockout switch and a bursty workload), represented as SANs. The *UltraSAN* software has been implemented in a modular fashion and is based on an X-window user interface.

The user interface consists of several modules which are unified via the control panel. For example, the SAN model represented in Figure 2 is first drawn using *sanedit* which is a SAN editor. The switching functions and workload characteristics, i.e., the various gate functions, activity definitions, and case probabilities (all as described in Section III), can be coded into the SAN model via sanedit. *UltraSAN* allows for hierarchical model construction via the composed model editor, also called *compedit*. The reward variables representing the performance measures of interest can be specified by the performability variable editor, also referred to as *varedit*.

As discussed with respect to Table 4 in Section III, *UltraSAN* also allows for parameterization of the model via the use of global variables. An editor called the study editor (*stdedit*) is used for this purpose. The resulting experiments from model parameterization can be organized into logical groups called studies. For example, a study may consist of a set of experiments in which only one of several global variables changes.

In order to obtain an analytic solution, *UltraSAN* first constructs the reduced base model (RBM) generator. The RBM generator is an executable program that when run creates a machine readable description of the Markov process underlying the knockout switch and bursty workload. Reference [19] discusses the details of the algorithms used for constructing

the RBM. After constructing the underlying stochastic processes, *UltraSAN* provides two solvers for steady-state results. They are: (a) a direct solver for relatively small models and (b) an iterative solver for large models (in the order of several hundreds of thousands of states). In this work, we use the iterative solver because of the largeness of the state-space size, which is illustrated in Table 7. The steady-state solvers provide information about the mean, variance, probability density function and probability distribution function of each of the performance variables, based on the steady-state state occupancy probabilities.

The ATM switch model, as discussed in Sections II and III, was represented by a discrete-time Markov chain (DTMC). It can, however, be easily shown that a DTMC with a single timed (deterministic) activity can be replaced by a CTMP with a single exponential activity. The equivalence between the two is obtained by re-writing the transition probability matrix, $\mathbf{P}$, of the DTMC in terms of the transition rate matrix, $\mathbf{Q}$, of the CTMP, and solving for the steady-state probability distribution vector. Thus, if the time associated with the single deterministically timed activity is t, we have $\mathbf{P} = \mathbf{Q} \cdot t + \mathbf{I}$, and the equivalence between the DTMC and CTMP is easily established.

This equivalence allows us to use *UltraSAN* to generate the corresponding Markov process and to obtain numerical results for CLP and distribution of output queue length. The state-space size of the resulting process is dependent on the number of input ports, buffer, and concentrator size. For example, the state-space size for a 16-input switch as a function of buffer size M with $L = 8$ is shown in Table 7. Observe the approximately linear growth in state-space size with increasing buffer size.

Upon generating the Markov process, the steady-state probability distribution vector is determined numerically using successive over-relaxation (SOR). The algorithm is stopped when the maximum difference between the most recent and previous iteration is less than 10^{-9}. The CLP and distribution of queue length are then obtained. The number of itera-

Buffer size (M)	*Number of states*
40	6103
80	12223
120	18343
200	30583
280	42823
360	55063

Table 7: Variation of state-space size with buffer capacity

tions to converge to the given accuracy is dependent on the model parameters (transition rates), and even for the largest buffer case, it was within minutes (< 25).

A variety of studies are conducted with varying burstiness, as defined by the parameters AF, BS, and q. While defining these parameters is easy, choosing realistic values is not. This is because of their dependency on: (a) the number and type of applications multiplexed on a port, (b) the port speed, and (c) the aggregate load as seen at an input port. Based on the relative mix of (a), (b), and (c), the overall traffic to the port may have AFs ranging from very low (peaked) to large (smoothed due to aggregation) values. A similar phenomenon with respect to the load as seen on a LAN is described in [8], where the authors discuss LAN workload characterization. Their results highlight the smoothening effect due to aggregation at high loads, while at low loads, the bursty nature of an individual source is reflected closely on the overall LAN traffic parameters. In addition, the expected burst size as seen by an input port is not simple to quantify and, as illustrated in [9], depends on the nature of the applications multiplexed.

Thus, since the BS and AF of an input port vary widely, we examine a wide range of these parameters, indicative of a mix of applications, while analyzing switch performance. The remainder of this section is devoted to discussing our results.

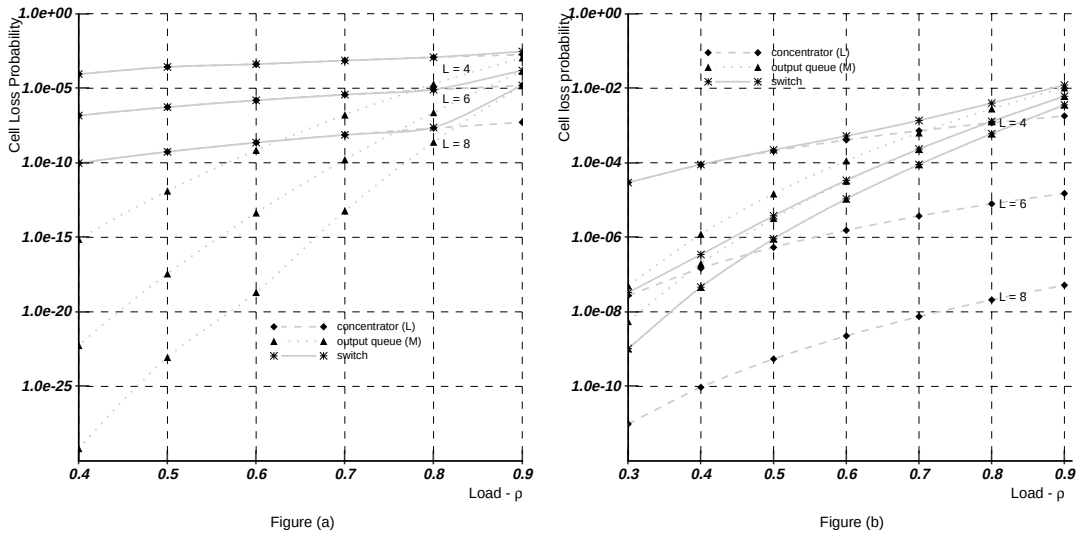

Figure 3: Effect of concentrator size on CLP with steady (Figure(a)) and bursty (Figure(b)) traffic

Role of concentrator and output queue with steady and bursty traffic Figures 3(a) and (b) display CLPs at the concentrator, output queue and switch with steady and bursty traffic, respectively. In Figure 3(b), the AFs are varied from 0.27 to 0.81 to vary ρ. BS is set to 8 and q to 0.07. The queue size is $M = L \times 5$ in both figures, i.e., when L equals 4, 6, and 8, M is 20, 30, and 40, respectively, as suggested in the original knockout switch proposal by [22]. The low AFs with low loads represent short sporadic traffic bursts representative of, for example, a small number of multiplexed user interactive data, and the higher AFs with higher loads represent, for example, bandwidth intensive file server and database applications.

The figures illustrate the change in the roles of the concentrator and output queue in their contributions to the overall CLP. With steady traffic, the concentrator, rather than the output queue, contributes significantly to the switch CLP for a wide range of loads (up to 0.8). With bursty traffic, however, the roles of the concentrator and output queue (in terms of being the bottleneck) reverse, based on AF and L. Specifically, when both the

concentrator size and AFs are small (in Figure 3(b) for $L = 4$, AF ≤ 0.55 and $\rho < 0.75$), the concentrator knocks out the clusters before they reach the output queue, becoming the bottleneck. However, with increasing AFs (decreasing burstiness), the output queue becomes the bottleneck (L=4, $AF > 0.55$ and $\rho \geq 0.75$). With a larger concentrator size ($L > 6$, which is the suggested size for the knockout switch), the concentrator knocks out much fewer cells than the output queue over the entire range of AFs. This makes the output queue rather than the concentrator the bottleneck with bursty traffic, an important difference in our results compared to those reported for steady traffic [22].

Effect of AF on CLP Figure 4 demonstrates the effect of AF on CLP. The AF is varied (0.2-0.8) for a fixed BS (8), and three ρ's (0.3, 0.5, and 0.8) to provide the effect of multiplexing different types and numbers of sources. The higher AFs reflect multiplexing a large number of small bandwidth applications, while lower AFs reflect multiplexing a small number of large bandwidth applications. The concentrator size in this and subsequent graphs is set to 8. (This is because $L = 8$ contributes negligibly to the overall CLP as seen from Figure 3(b).) Further, unless otherwise mentioned, M is set to 40.

Observe the rapid deterioration in CLP with increasing burstiness (decreasing AFs) despite a fixed average load, demonstrating the significance of traffic bursts on CLP. The results also indicate that low AFs do not always imply poor CLP, since low AFs with low loads are not as detrimental as low AFs and high loads. Note also the widely differing CLPs (QoS) from 10^{-2} to 10^{-8} with low AF (0.2) and three different ρ's. This suggests the necessity of efficient policing and CAC techniques, which react based on the AF-load pair combination, rather than policing and admitting on AF or load alone. For example, with a low AF, larger swings/deviations may be permitted with low average loads.

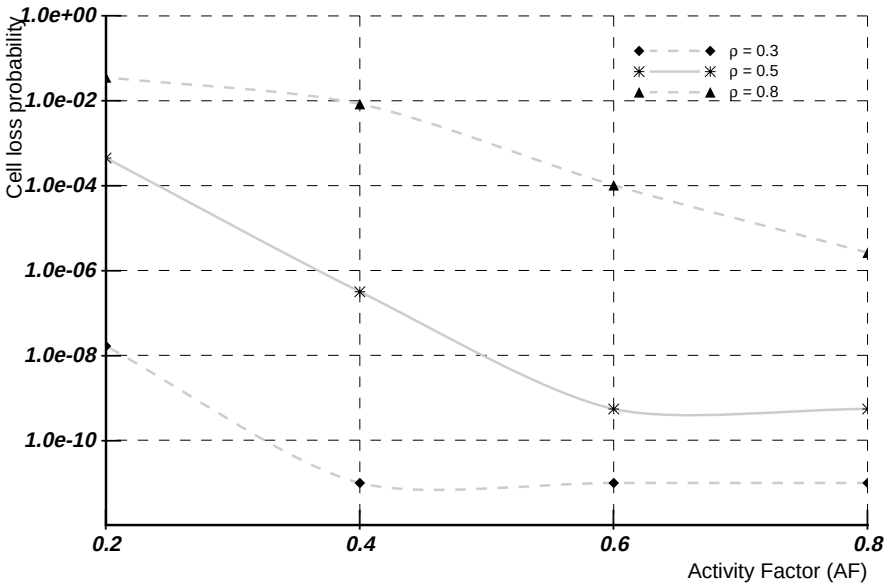

Figure 4: Effect of increasing AF on CLP

Effect of E_{active} on CLP Figure 5 demonstrates the effect of varying ON times on the CLP. We fix ρ at 0.8 and use three AFs (0.3, 0.5, and 0.8) and vary the mean ON times to examine the effect of varying burst sizes (BS), hence varying congestion levels for a fixed *average* load with each AF. The mean ON times are varied from 48-768, 80-1280, and 128-2048 cells, respectively, for the three AFs, resulting in an average BS ranging from 8-128.

Apart from the expected increase in CLP with increasing BS, observe the detrimental effect of smaller AFs (with the same BS and load) on the CLP. Observe also that the change in CLP between BS 8 and 128 is smaller for lower AFs (0.3) than for higher AFs (0.8). This is because low AFs imply less smoothening and hence, regardless of the active time, exhibit high CLP. Next, as mean ON times increase, the performance, even with higher AFs, becomes poor due to increased spurts of congestion, despite the same *average* load. Finally, the results for lower average loads each with the three different AFs exhibit similar trends but are lower in magnitude (lower CLP) as expected and are therefore not

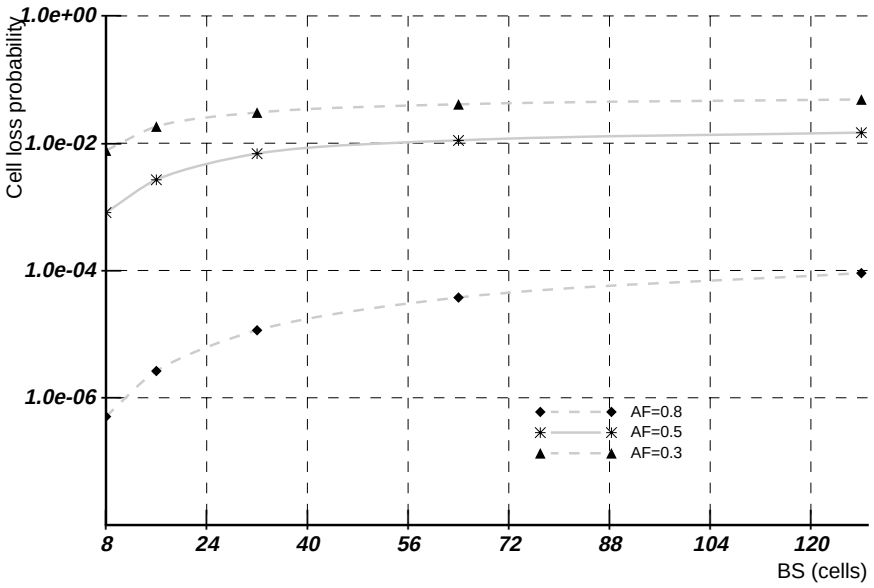

Figure 5: Effect of increasing E_{active} on CLP

included. All these indicate that switch performance at a given average load is largely dictated by the combination of AF and BS. Thus, characterization of switch performance at different average loads alone with bursty traffic is incomplete, and attention has to be paid to the burst parameters as well.

Effect of buffer size In Figure 6, which displays the effect of increasing buffer size on CLP (QoS), ρ is fixed at 0.8, BS at 8, and three AFs (0.8, 0.5, and 0.3) are used. CLP improvement with increasing buffer size is fairly good for high AFs (0.8). However, when AFs become small (≤ 0.3), the curves exhibit a much steeper slope, indicative of less gains. The load, however, seen at the switch in all three cases is the same. Again, this illustrates the significance of bursty traffic and highlights the fact that knowledge of average load alone is insufficient while analyzing performance and dimensioning switches. It indicates the necessity of combining prioritized discarding and congestion control strategies along with buffer sizing to guarantee acceptable QoS with bursty traffic, especially critical with

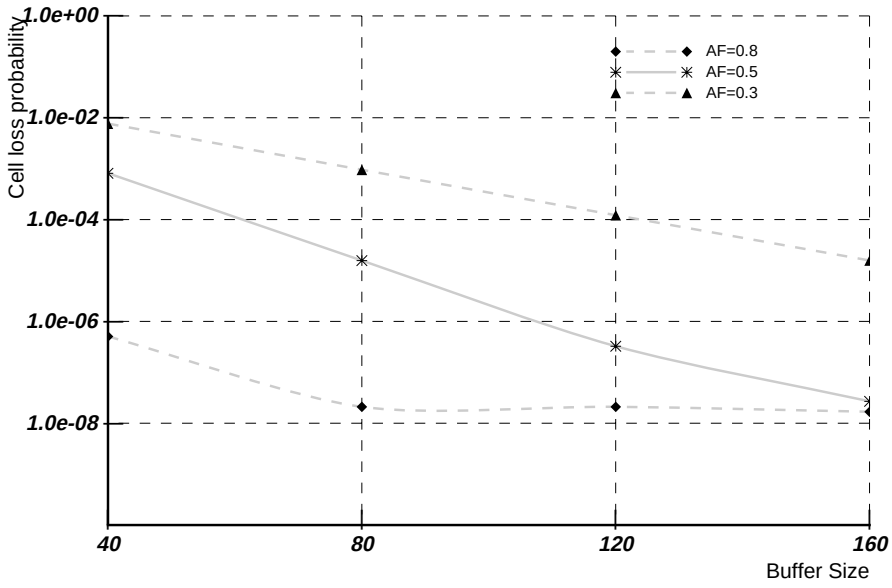

Figure 6: Effect of buffer size on CLP with bursty traffic

low AFs and medium to high loads. Finally, the shapes of the response curves with lower average loads (which are not included due to brevity), follow that of their higher load counterparts, but exhibit less steep slopes even with low AFs.

Effect of AF and E_{active} on the distribution of queue length We next investigate the distribution of queue length with steady and bursty traffic in Figures 7 through 10 with $\rho = 0.8$. With steady traffic (Figure 7), there is a mass build up at the origin, with a quick decay subsequently. With bursty traffic, the behavior is very different, depending on the AF and BS. (Note the change in scales between Figure 7 and Figures 8 through 10.) With low AFs, the distribution of queue length exhibits a bimodal behavior, with mass build up at the extreme positions (origin and queue-fill). Further, for the same average load and small AFs, the bimodal behavior becomes more distinct as the BS increases (Figures 8(a) and 8(b)). However, as the AFs increase (≥ 0.5), the intensity of the bimodal behavior decreases despite large changes in BS (8 and 128 or ON times of 80 and 1280, Figures 9(a)

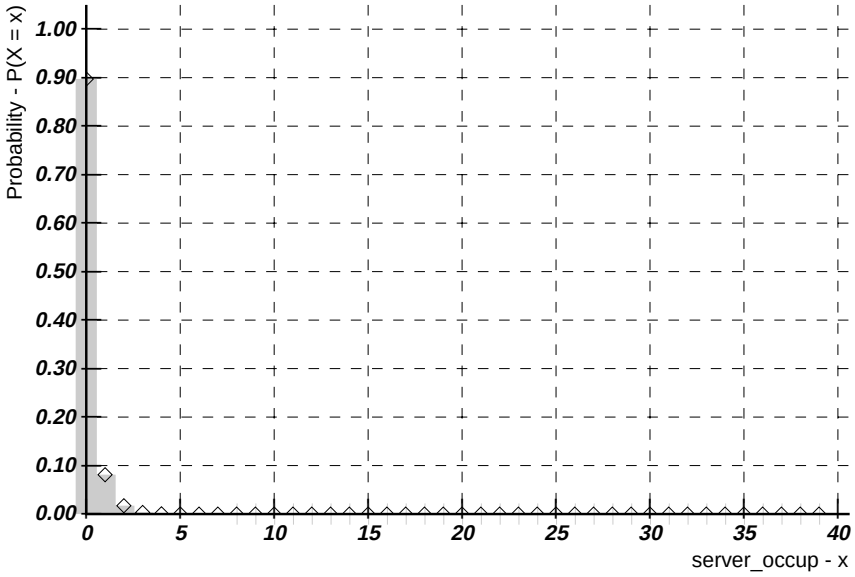

Figure 7: Density function of queue length with steady traffic

and 9(b)). Finally for large AFs (0.8) (Figures 10(a) and 10(b)), the distribution of queue length does not exhibit bimodal behavior even when ON times change from 128 to 2048 cells, but almost resembles that with steady traffic, just as expected, since larger AFs imply smoother (steadier) traffic.

The bimodal behavior implies that with high probabilities, the queue is either empty or full. Thus, the average queue length in such a scenario is not very meaningful, since if the queue is full, it will likely remain so for quite some time. This implies that arrivals during the time to drain the queue will be lost successively, causing a high CLP (severe degradation in QoS). However, for the same average queue length, if the distribution was not bimodal but instead a function decaying quickly to negligible values at queue fill (as in Figures 7, 10(a), and 10(b)), the CLP (hence QoS) degradation would not be as bad.

Therefore, bimodal behavior, especially when pronounced, produces detrimental effects. Observe again the importance of switch characterization under varying burst intensities, since the distribution of queue length varies significantly for different AF and BS combina-

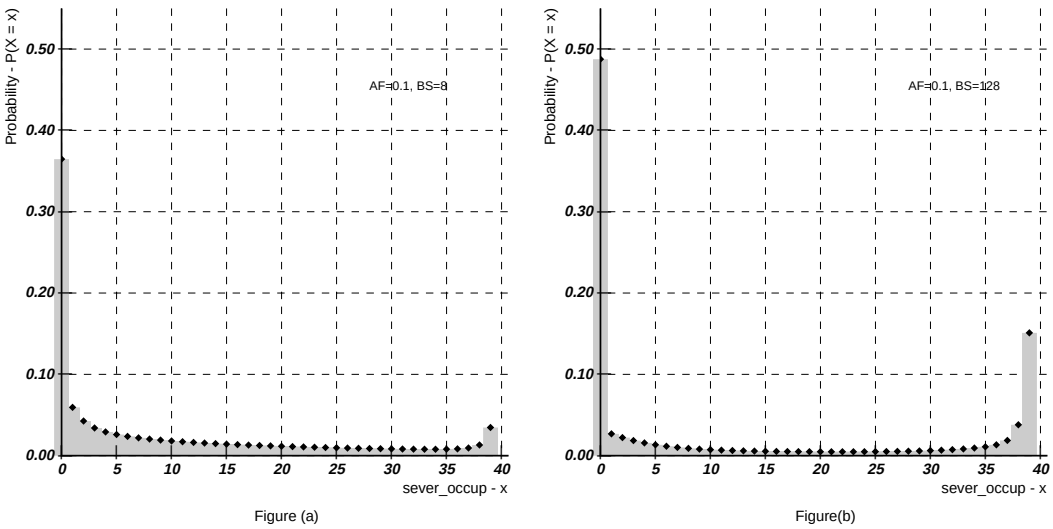

Figure 8: Density function of queue length with AF=0.1 and BS=8 (Figure (a)) and AF=0.1, BS=128 (Figure (b))

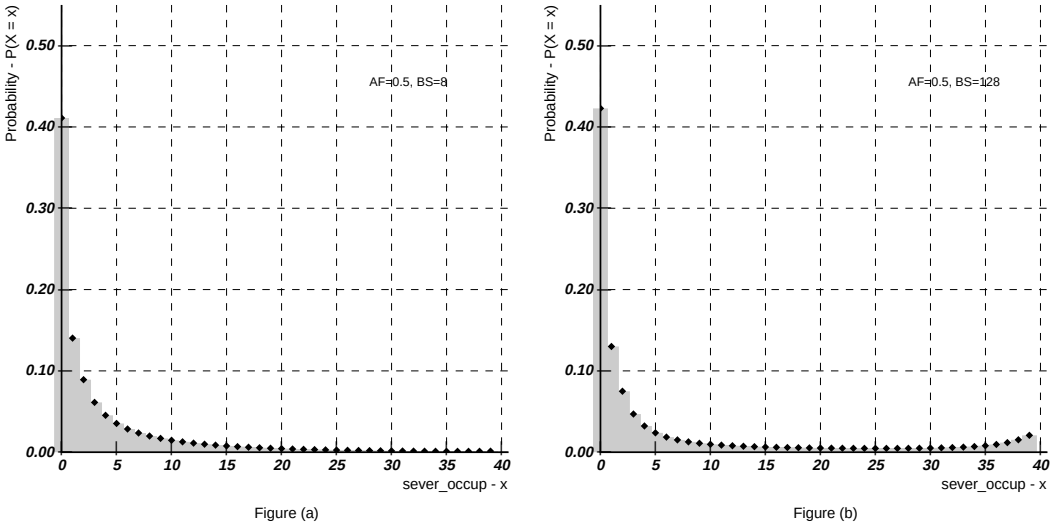

Figure 9: Density function of queue length with AF=0.5 and BS=8 (Figure (a)) and AF=0.5, BS=128 (Figure (b))

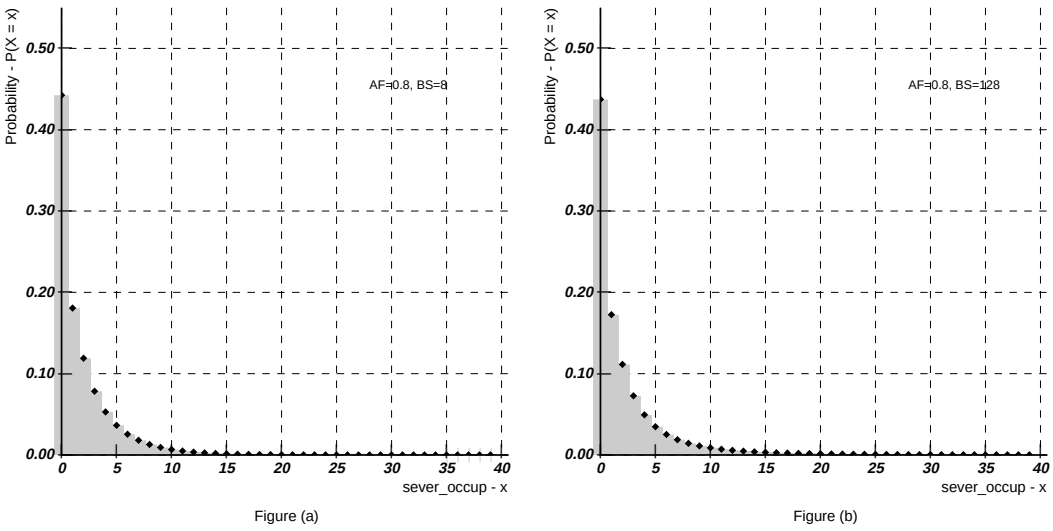

Figure 10: Density function of queue length with AF=0.8 and BS=8 (Figure (a)) and AF=0.8, BS=128 (Figure (b))

tions despite a fixed *average* load. Finally, notice that except for a particularly bad case (very small AF (≤ 0.1), large ON time (≥ 2048), and fairly high average load ($\rho = 0.8$)), the knockout mechanism is seen to provide good CLP performance over the range of burst parameters and loads studied.

V Conclusions

Our contributions are two-fold. First, we demonstrate the appropriateness of stochastic activity network models for analytically predicting the performance of ATM switch architectures. SANs help capture the details of the FPS and workload in a compact manner, thus enabling us to accurately characterize its functioning. They also help in the automatic generation and solution of the detailed underlying Markov process in reasonable time. Since the size of these processes for reasonable switch dimensions is on the order of tens of thousands of states, hand construction and solution of such processes become not only unreasonable, but also distract the modeler from incorporating practical design constraints and discard-

ing algorithms. Our results, however, show that it is indeed possible to model and analyze realistic switch architectures and workloads as SANs.

Second, we examine the sensitivity and robustness of the knockout mechanism with bursty traffic, and illustrate the importance of analyzing switch performance not just via the average offered load, but also as a function of the burst parameters. With respect to robustness and sensitivity, we observe the following. While the overall CLP is more sensitive to the concentrator rather than the output queue size with steady traffic, the situation reverses with bursty traffic and concentrator size > 6 (the practical suggested size) and over a wide range of burst parameters. Next, either low AFs or large BSs alone do not necessarily imply poor performance, and the knockout mechanism is seen to be robust over the range of burst parameters studied. It is the combination of high loads with low AFs (e.g., multiplexing a small number of bandwidth intensive bursty applications) or large burst sizes with very low AFs that cause rapid CLP degradation.

Regarding CLP improvement with respect to buffer size, though the logarithm of CLP decreases linearly with an increasing buffer size, the gains with larger buffers for the same average load become significantly less with lower AFs (peaked traffic). The distribution of the queue length with steady and bursty traffic confirmed the existence of a bimodal behavior with bursty traffic. CLP degradation especially with pronounced bimodal behavior was pointed out, illustrating that the average queue length under such conditions is not a meaningful measure.

All of the above reiterate the need for a combination of efficient rate regulation, prioritized discarding, and congestion control techniques, along with buffer sizing to obtain acceptable QoS. In addition, while it is important in general for sources with low AFs (highly bursty) to adhere to their negotiated burst parameters, a larger deviation may be permitted even with these low AFs, depending on the average load seen at the port.

Acknowledgements: The authors would like to thank the editor and anonymous reviewers for their helpful comments.

REFERENCES

[1] H. Ahamadi and W. E. Denzel, "A survey of modern high-performance switching techniques," *IEEE Journal on Selected Areas in Communications*, vol. 7, no. 7, pp. 1091-1103, 1989.

[2] W. E. Denzel, A. P. J. Engbersen, I. Iliadis, and G. Karisson, "A highly modular packet switch for Gb/s rates," *Proceedings of the XIV International Switching Symposium*, vol. 2, A8.3, pp. 236-240, 1992.

[3] K. Y. Eng, M. G. Hluchyj, and Y. S. Yeh, "A knockout switch for variable-length packets," *IEEE Journal on Selected Areas in Communications*, vol. 5, no. 9, pp. 1426-1435, 1987.

[4] J. B. Evans, E. Duron, and Y. Wang, "Analysis and implementation of a priority knockout switch," *IEEE INFOCOM*, pp. 1090-1106, 1993.

[5] D. P. Heyman, A. Tabatabai, and T. V. Lakshman, "Statistical analysis and simulation study of video teleconference traffic in ATM networks," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 2, no. 1, pp. 49-59, 1992.

[6] D. P. Heyman and T. V. Lakshman, "Source models for VBR broadcast-video traffic," *INFOCOM*, pp. 664-671, 1994.

[7] M. J. Karol, M. G. Hluchyj, and S. P. Morgan, "Input versus output queueing on a space-division packet switch," *IEEE Transactions on Communications*, vol. 35, no. 12, pp. 1347-1356, 1987.

[8] K. M. Khalil, K. Q. Lue, and D. V. Wilson, "LAN traffic analysis and workload characterization," *Proceedings of the 15th Conference on Local Computer Networks*, pp. 112-122, 1990.

[9] K. M. Khalil and Y. S. Sun "The effect of bursty traffic on the performance of local area networks," *GLOBECOM*, pp. 597-603, 1992.

[10] W. E. Leland, M. S. Taqqu, W. Willinger, and D. V. Wilson, "On the self-similar nature of ethernet traffic (Extended Version)," *IEEE/ACM Transactions on Networking*, vol. 2, no. 1, pp. 1-15, 1994.

[11] S-Q. Li and J. W. Mark, "Traffic characterization for integrated services networks," *IEEE Transactions on Communications*, vol. 38, no. 8, pp. 1231-1243, 1990.

[12] J. F. Meyer, A. Movaghar, and W. H. Sanders, "Stochastic activity networks: Structure, behavior, and, application," *Proceedings of the International Workshop on Timed Petri nets*, pp. 106-115, 1985.

[13] A. Movaghar and J. F. Meyer, "Performability modeling with stochastic activity networks," *Proceedings of the Real-Time Systems Symposium*, pp. 215-224, 1984.

[14] I. Norros, "On the use of fractional brownian motion in the theory of connectionless networks," *IEEE Journal on selected areas in Communications*, vol. 13, no. 6, pp. 953-962, 1995.

[15] A. Pattavina, "Nonblocking architectures for ATM switching," *IEEE Communications Magazine*, pp. 91-95, February, 1993.

[16] J. M. Pitts, Z. Sun, J. P. Cosmas, and E. M. Scharf, "Burst-Level teletraffic modelling: Applications in broadband network studies," *Third IEE Conference on Telecommunications*, pp. 348-352, 1991.

[17] J. W. Roberts and A. Gravey, "Recent results on B-ISDN/ATM traffic modelling and performance analysis - A review of ITC 13 papers," *GLOBECOM*, pp. 1325-1330, 1991.

[18] W. H. Sanders, W. D. Obal, M. A. Qureshi, and F. K. Widjanarko, "The *UltraSAN* modeling environment," *Performance Evaluation Journal, Special Issue on Performance Modeling Tools*, vol. 24, no. 1, pp. 89-115, 1995.

[19] W. H. Sanders and J. F. Meyer, "Reduced base model construction methods for stochastic activity networks," *IEEE Journal on Selected Areas in Communications*, Vol. 9, pp. 25-36, January 1991.

[20] F. A. Tobagi, "Fast packet switch architectures for broadband integrated services digital network," *Proceedings of the IEEE*, vol. 78, no. 1, pp. 133-167, 1990.

[21] G. M. Woodruff and R. Kositpaiboon, "Multimedia traffic management principles for guaranteed ATM network performance," *IEEE Journal on Selected Areas in Communications*, vol. 8, no. 3, pp. 437-446, 1990.

[22] Y. S. Yeh, M. G. Hluchyj, and A. S. Acampora, "The knockout switch: A simple, modular architecture for high-performance packet switching," *IEEE Journal on Selected Areas in Communications*, vol. 5, no. 8, pp. 1274-1283, 1987.

[23] H. Yoon, M. T. Liu, and K. Y. Lee, "The knockout switch under nonuniform traffic," *GLOBECOM*, pp. 1628-1634, 1988.