Modelling of Packet Loss in an Asynchronous Packet Switch using PEPA

Wim Vanderbauwhede
Department of Computing Science
University of Glasgow
17 Lilybank Gardens, G12 8QQ
Email: wim@dcs.gla.ac.uk

Abstract

We present a PEPA model for the packet loss of an asynchronous packet switch. The model uses two-state traffic an an explicit enumeration of the state space. We show that the PEPA model results match the results obtained using discrete-event simulation.

I. ASYNCHRONOUS PACKET SWITCH

Packet switches switch data packets from a given ingress port to an egress port determined by information in the packet header. Asynchronous packet switches (Figure 1) don't assume synchronisation between arrivals of packets on different ingress ports, and don't attempt to synchronise the forwarding of packets to the egress ports. This type of packet switch is being investigated for high-speed optical core networks [1] and on-chip networks [2].
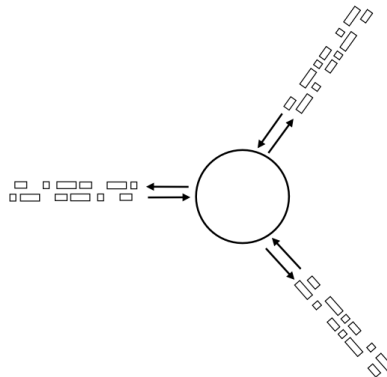

Fig. 1. Asynchronous packet switch

A. Contention and buffering

- Contention occurs when two or more packets want to occupy the same destination port.
- Buffering is used to resolve contention: one of the packets is forwarded to the egress port, the other is stored in a buffer cell. As the number of buffer cells is finite, the buffer can overflow, causing packet loss.

B. Properties

The switch model used in this work has following specific properties (Figure 2):

- Store-and-forward architecture: packet is stored in buffer cell, then forwarded. This as opposed to "cut-through" switches, where the packet does not need to be stored before being forwarded.
- A packet must have left the buffer cell completely before another packet can occupy the same cell.
- Simultaneous egress from buffers to different egress ports is possible (transparent).
- The actual switch fabric is a black box. Furthermore, for any incoming packet, all egress destinations have the same likelihood.
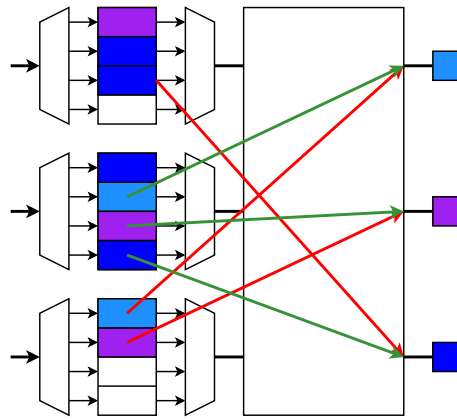
Fig. 2. Packet switch schematic

C. Steady-State Packet Loss

The most important performance indicator for an asynchronous packet switch with buffering is the steady-state packet loss. This is the average packet loss incurred by a switch for traffic with a fixed average load. The packet loss depends on the load (ingress rate versus maximum rate) and the buffer depth (number of cells per buffer)., but also on traffic distribution and switch architecture. PEPA is useful to analyse the former two influences, for the latter two a DES is required.

II. THE PEPA MODEL FOR THE SWITCH

A. Queue Model for Packet Switch and translation to PEPA

The switch can be modelled as a system of c interacting $M/M/c/N$ queues (Fig. 3):
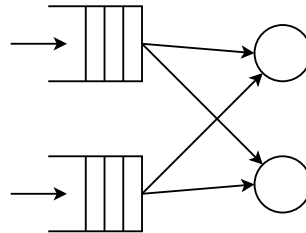

Fig. 3. Queue model for the packet switch

Our PEPA model for this system of interacting queues is presented in Figure 4. G is the traffic generator, Q the buffer queue and M the output multiplexer, i.e. the actual switching part of the packet switch. // indicates components in parallel (empty interaction) and the arrow →indicates the direction of active/passive interaction.
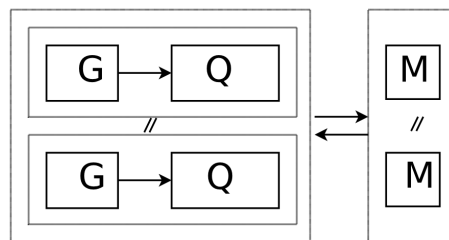

Fig. 4. Organisation of the PEPA model

B. Two-state traffic generator

The traffic generator G

- Models packet traffic for packets with variable length and inter-arrival time
- By two-state traffic we mean traffic where the packet length distribution is modelled by the on state and the gap length distribution by the off state.
- We use following definitions:
  τ_{on}, τ_{off}: time period during which the source is *on* resp. *off*.
  $\lambda_{on} = 1/\tau_{on}$, $\lambda_{off} = 1/\tau_{off}$: the corresponding rates
- Model:

$$G = (off, \lambda_{off}).(on, \lambda_{on}).G$$
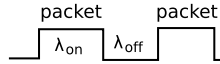
C. A single queue with two-state traffic

Our main aim is to model packet loss. To do so, we introduce the concept of "drop" states. This is the state which the system enters after a packet has been dropped. The main complication of this concept is that it introduces an extra drop state for every reachable state.

The way to calculate the drop probability is quite straightforward: it is the probability that a packet is being dropped over the probability that a packet is being received. The latter is the total probability multiplied by the load (on/(on+off)).

To build the actual model, we proceed in four steps:

1) Single ingress, single egress, no drop states
2) Single ingress, single egress, with drop states
3) Single ingress, multiple egress, no drop states
4) Single ingress, multiple egress, with drop states

1) Single ingress, single egress, no drop states: Because the traffic is asynchronous, arriving and leaving packets can overlap in time. In its simplest form, we have this behaviour: buffer system in state i; a packet arrives, starts filling the buffer; a packet leaves, starts emptying the buffer. This behaviour leads to 4 distinct states, for which we introduce following notation:

$$Q_i^{+j}_{-k} \ , i \in \{0, ..., N\}; j \in \{0, 1\}; k \in \{0, 1\}$$
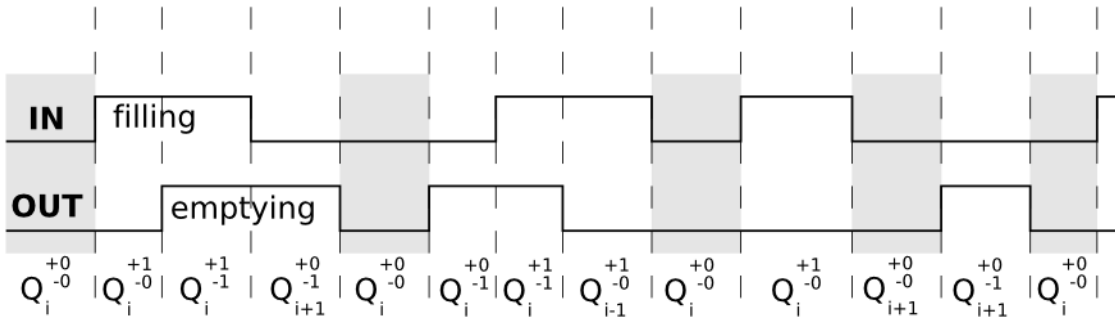

Fig. 5.   Illustration of states in a queue with 1 ingress port and 1 egress port and on/off traffic

We refer to Q_i as the "base state", i.e. regardless of the packets arriving or leaving, the buffer occupancy is i. The "base state" is actually a set of $j.k$ states. The state transitions are caused by the arrival of a packet or the arrival of a signal telling a packet to leave. We indicate the presence/absence of a packet at the ingress/egress port as $\{on, off\}_{\{in, out\}}$ with corresponding rates $\lambda_{\{on, off\}, \{in, out\}}$.

Note that $\lambda_{on,in}$ the rate is at which a packet fills the buffer, and therefore $\lambda_{on,in} \equiv \lambda_{on,out}$; furthermore, $\lambda_{off,out}$ relates to the delay between moment when the egress port becomes free and the moment when a packet starts leaving the buffer. With these conventions, the PEPA model becomes:

$0 < i < N$

$$Q_i{}^{+0}_{-0} = (off_{in}, \lambda_{off,in}).Q_i{}^{+1}_{-0} + (off_{out}, \lambda_{off,out}).Q_i{}^{+0}_{-1}$$

$$Q_i{}^{+1}_{-0} = (on_{in}, \lambda_{on,in}).Q_{i+1}{}^{+0}_{-0} + (off_{out}, \lambda_{off,out}).Q_i{}^{+1}_{-1}$$

$$Q_i{}^{+0}_{-1} = (off_{in}, \lambda_{off,in}).Q_i{}^{+1}_{-1} + (on_{out}, \lambda_{on,out}).Q_{i-1}{}^{+0}_{-0}$$

$$Q_i{}^{+1}_{-1} = (on_{in}, \lambda_{on,in}).Q_{i+1}{}^{+0}_{-1} + (on_{out}, \lambda_{on,out}).Q_{i-1}{}^{+1}_{-0}$$

In the empty state, no packets can leave:

$i = 0$

$$Q_0{}^{+0}_{-0} = (off_{in}, \lambda_{off,in}).Q_0{}^{+1}_{-0}$$

$$Q_0{}^{+1}_{-0} = (on_{in}, \lambda_{on,in}).Q_1{}^{+0}_{-0}$$

In the full state, new packets arriving don't change the base state; otherwise, this state is identical to the lower states.

$i = N$

$$Q_N{}^{+0}_{-0} = (off_{in}, \lambda_{off,in}).Q_N{}^{+1}_{-0} + (off_{out}, \lambda_{off,out}).Q_N{}^{+0}_{-1}$$

$$Q_N{}^{+1}_{-0} = (on_{in}, \lambda_{on,in}).Q_N{}^{+0}_{-0} + (off_{out}, \lambda_{off,out}).Q_N{}^{+1}_{-1}$$

$$Q_N{}^{+0}_{-1} = (off_{in}, \lambda_{off,in}).Q_N{}^{+1}_{-1} + (on_{out}, \lambda_{on,out}).Q_{N-1}{}^{+0}_{-0}$$

$$Q_N{}^{+1}_{-1} = (on_{in}, \lambda_{on,in}).Q_N{}^{+0}_{-1} + (on_{out}, \lambda_{on,out}).Q_{N-1}{}^{+1}_{-0}$$

2) Single ingress, single egress, with drop states: We are interested in the packet loss in steady state. We calculate this as sum of the probabilities of being in a state where the packet is being dropped. To do so, we must add "drop states". This is, surprisingly, fairly straightforward. The full state leads to a drop state on arrival of a packet:

$i = N$

$$Q_N{}^{+0}_{-0} = (off_{in}, \lambda_{off,in}).Q_{N,d}{}^{+1}_{-0} + (off_{out}, \lambda_{off,out}).Q_N{}^{+0}_{-1}$$

$$Q_N{}^{+0}_{-1} = (off_{in}, \lambda_{off,in}).Q_{N,d}{}^{+1}_{-1} + (on_{out}, \lambda_{on,out}).Q_{N-1}{}^{+0}_{-0}$$

All base states (except the empty state) gain 2 drop states, which express that while the packet is being dropped, one or more packets can start/stop leaving the buffer, leading to a lower base state.

$0 < i \le N$

$$Q_{i,d}^{\substack{+1\\-0}} = (on_{in},\lambda_{on,in}).Q_i^{\substack{+0\\-0}} + (off_{out},\lambda_{off,out}).Q_{i,d}^{\substack{+1\\-1}}$$

$$Q_{i,d}^{\substack{+1\\-1}} = (on_{in},\lambda_{on,in}).Q_i^{\substack{+1\\-1}} + (on_{out},\lambda_{on,out}).Q_{i-1,d}^{\substack{+1\\-0}}$$

In the empty state, no packets can leave, so this state becomes simply:

$i = 0$

$$Q_{i,d}^{\substack{+1\\-0}} = (on_{in},\lambda_{on,in}).Q_i^{\substack{+0\\-0}}$$

3) Single ingress, multiple egress, no drop states: To create a true concurrent-egress model, we extend the set of substates for leaving: $k \in \{0, ..., M_{out}\}$. This leads to a considerable increase of the number of substates:

Let $c = min(N, M_{out})$ and $\lambda_{\{on,off\},out,k} = k.\lambda_{\{on,off\},out}$. The previous equations change to:

$0 < i < N, 0 < k < c$

$$Q_i^{\substack{+0\\-k}} = (off_{in},\lambda_{off,in}).Q_i^{\substack{+1\\-k}} + (on_{out},\lambda_{on,out,k}).Q_{i-1}^{\substack{+0\\-(k-1)}} + (off_{out},\lambda_{off,out,c-k}).Q_i^{\substack{+0\\-(k+1)}}$$

$$Q_i^{\substack{+1\\-k}} = (off_{in},\lambda_{off,in}).Q_{i+1}^{\substack{+0\\-k}} + (on_{out},\lambda_{on,out,k}).Q_{i-1}^{\substack{+1\\-(k-1)}} + (off_{out},\lambda_{off,out,c-k}).Q_i^{\substack{+1\\-(k+1)}}$$

With similar equations for the limiting cases:

$i = 0 \Rightarrow k = 0$:

$$Q_i^{\substack{+0\\-0}} = (off_{in},\lambda_{off,in}).Q_i^{\substack{+1\\-0}}$$

$$Q_i^{\substack{+1\\-0}} = (off_{in},\lambda_{off,in}).Q_{i+1}^{\substack{+0\\-0}}$$

$0 < i < N, k = 0$:

$$Q_i^{\substack{+0\\-0}} = (off_{in},\lambda_{off,in}).Q_i^{\substack{+1\\-0}} + (off_{out},\lambda_{off,out,c-k}).Q_i^{\substack{+0\\-1)}}$$

$$Q_i^{\substack{+1\\-0}} = (off_{in},\lambda_{off,in}).Q_{i+1}^{\substack{+0\\-0}} + (off_{out},\lambda_{off,out,c-k}).Q_i^{\substack{+1\\-1}}$$

$0 < i < N, k = c$:

$$Q_i\overset{+0}{}\overset{}{_{-c}} = (off_{in}, \lambda_{off,in}).Q_i\overset{+1}{}\overset{}{_{-c}} + (on_{out}, \lambda_{on,out,k}).Q_{i-1}\overset{+0}{}\overset{}{_{-(c-1)}}$$

$$Q_i\overset{+1}{}\overset{}{_{-c}} = (off_{in}, \lambda_{off,in}).Q_{i+1}\overset{+0}{}\overset{}{_{-c}} + (on_{out}, \lambda_{on,out,k}).Q_{i-1}\overset{+1}{}\overset{}{_{-(c-1)}}$$

$i = N, 0 < k < c$:

$$Q_N\overset{+0}{}\overset{}{_{-k}} = (off_{in}, \lambda_{off,in}).Q_N\overset{+1}{}\overset{}{_{-k}} + (on_{out}, \lambda_{on,out,k}).Q_{N-1}\overset{+0}{}\overset{}{_{-(k-1)}} + (off_{out}, \lambda_{off,out,c-k}).Q_N\overset{+0}{}\overset{}{_{-(k+1)}}$$

$$Q_N\overset{+1}{}\overset{}{_{-k}} = (off_{in}, \lambda_{off,in}).Q_N\overset{+0}{}\overset{}{_{-k}} + (on_{out}, \lambda_{on,out,k}).Q_{N-1}\overset{+1}{}\overset{}{_{-(k-1)}} + (off_{out}, \lambda_{off,out,c-k}).Q_N\overset{+1}{}\overset{}{_{-(k+1)}}$$

$i = N, k = 0$:

$$Q_N\overset{+0}{}\overset{}{_{-0}} = (off_{in}, \lambda_{off,in}).Q_N\overset{+1}{}\overset{}{_{-0}} + (off_{out}, \lambda_{off,out,c-k}).Q_N\overset{+0}{}\overset{}{_{-1}}$$

$$Q_N\overset{+1}{}\overset{}{_{-0}} = (off_{in}, \lambda_{off,in}).Q_N\overset{+0}{}\overset{}{_{-0}} + (off_{out}, \lambda_{off,out,c-k}).Q_N\overset{+1}{}\overset{}{_{-1}}$$

$i = N, k = c$:

$$Q_N\overset{+0}{}\overset{}{_{-c}} = (off_{in}, \lambda_{off,in}).Q_N\overset{+1}{}\overset{}{_{-c}} + (on_{out}, \lambda_{on,out,k}).Q_{N-1}\overset{+0}{}\overset{}{_{-(c-1)}}$$

$$Q_N\overset{+1}{}\overset{}{_{-c}} = (off_{in}, \lambda_{off,in}).Q_N\overset{+0}{}\overset{}{_{-c}} + (on_{out}, \lambda_{on,out,k}).Q_{N-1}\overset{+1}{}\overset{}{_{-(c-1)}}$$

4) Single ingress, multiple egress, with drop states: Adding the dropstates is again straightforward:

$0 < i \leq N, 0 < k < c$

$$Q_{i,d}\overset{+1}{}\overset{}{_{-k}} = (off_{in}, \lambda_{off,in}).Q_i\overset{+0}{}\overset{}{_{-k}} + (on_{out}, \lambda_{on,out,k}).Q_{i-1,d}\overset{+1}{}\overset{}{_{-(k-1)}} + (off_{out}, \lambda_{off,out,c-k}).Q_{i,d}\overset{+1}{}\overset{}{_{-(k+1)}}$$

$0 < i \leq N, k = 0$

$$Q_{i,d}\overset{+1}{}\overset{}{_{-0}} = (off_{in}, \lambda_{off,in}).Q_i\overset{+0}{}\overset{}{_{-0}} + (off_{out}, \lambda_{off,out,c-k}).Q_{i,d}\overset{+1}{}\overset{}{_{-1}}$$

$0 < i \leq N, k = c$

$$Q_{i,d}\overset{+1}{}\overset{}{_{-c}} = (off_{in}, \lambda_{off,in}).Q_i\overset{+0}{}\overset{}{_{-c}} + (on_{out}, \lambda_{on,out,k}).Q_{i-1,d}\overset{+1}{}\overset{}{_{-(c-1)}}$$

This model's results are rather close to those of the rate-based model with egress rates scaled by M_{out}: $\lambda_{\{on,off\},M_{out}} = M_{out}.\lambda_{\{on,off\},out})$. Crucially, both rates must be scaled.

D. A more complex Nxc switch with buffers

To combine N of the above queues Qc with c outputs into a switch, we first introduce a multiplexer M. The model is identical to that of the traffic generator G. The final switch consists of N cooperations of G and Q_c in parallel, cooperating with c multiplexers in parallel

$$S_c = G < on_{in}, off_{in} > Q_c$$
$$Switch = (S_c \parallel ... \parallel S_c) < on_{out}, off_{out} > (M \parallel ... \parallel M)$$

E. A simple rate-based NxM switch with buffers

To benchmark the developed model, we compare it to a very simple rate-based model. The results should at least show the same overall tendencies.

Define a traffic generator generating packets at rate λ :

$$G = (in, \lambda).G$$

And a multiplexer taking in packets at rate μ, defined as $\frac{\lambda}{\rho}$:

$$M = (out, \mu).M$$

The buffer model is:

$$Q_0 = (in, \lambda).Q_1$$
$$Q_i = (in, \top).Q_{i+1} + (out, \mu).Q_{i-1} \quad , 0 < i < N$$
$$Q_N = (in, \top).Q_{Nd} + (out, \mu).Q_{N-1}$$
$$Q_{Nd} = (in, \top).Q_{Nd} + (out, \mu).Q_{N-1}$$

III. TOOLCHAIN FOR THIS WORK

The toolchain used for this work is the following:

- PEPA Workbench to calculate the transition rate matrix (TRM). We used a slightly patched version of v0.9.4b
- MatLab to calculate the steady-state solution. For our PEPA models, MatLab performed better than the Workbench.
- A Perl script to drive the simulation:
  - generate the input file for the PEPA Workbench & run
  - generate a MatLab file & run
  - calculate the packet loss probability from the steady-state
- The Simulation::Automate Perl package to automate the DOE. This package is a generic DOE automation framework, greatly simplyfing the task of running and postprocessing simulations with a large parameter space. It can be downloaded freely from CPAN [3].

IV. SOME RESULTS

A. Influence of the Signalling Delay

In a realistic switch model, the signalling delay ($\frac{1}{\lambda_{off,out}}$) would be constant. In PEPA, all delays have a negative exponential distribution. However, we can see from Figure 6 that the packet loss probability is independent of the signalling rate for signalling rates more that 10 times the data rate.
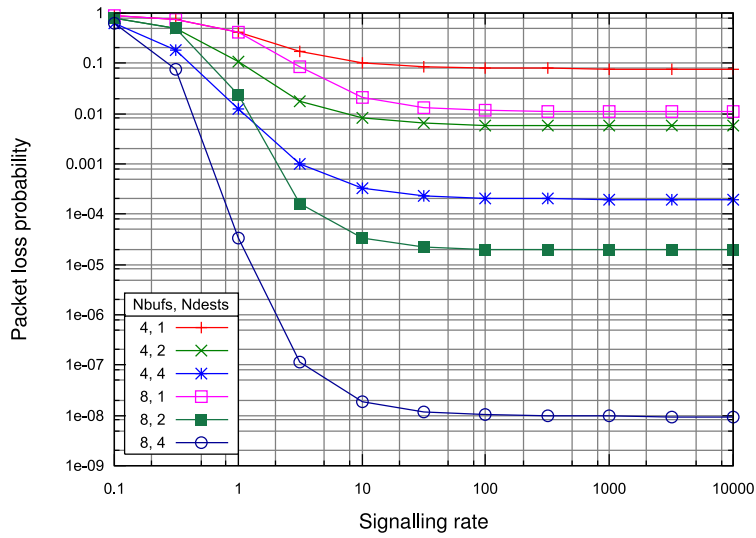
Fig. 6. Influence of the signalling delay on the packet loss probability

B. Comparison with Discrete Event Simulator

The main motivation for this work was the validation of results obtained from discrete-event simulations (DES). The DES allow a more realistic model for the packet traffic, but the simulator is very complex, which increases the chance of bugs. Therefore, validation with an analytical model is an important. We configured the DES with a two-state negative exponential traffic distribution as used in this work. Figure 7 shows that the results are in good agreement. The remaining discrepancies are due to the fact that PEPA cannot model constant delays and the fact that the DES cannot model an exact negative exponential distribution: real-life packets must have a non-zero minimum lenght and a finite maximum length.
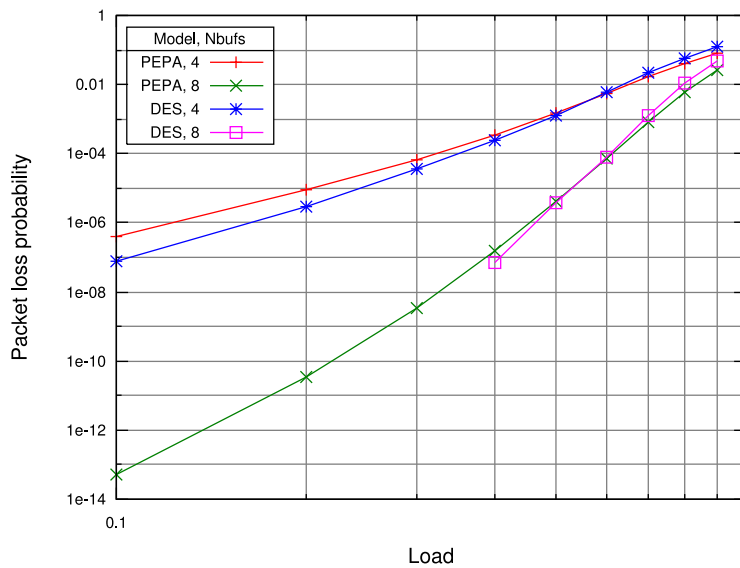

Fig. 7. Comparison of packet loss from PEPA model and from DES

C. Comparison with Rate-based Model

Figure 8 shows the comparison of the developed switch model with the simple rate based model presented in II-E. Both models display similar trends, but the rate-based model will always overestimate the packet loss.
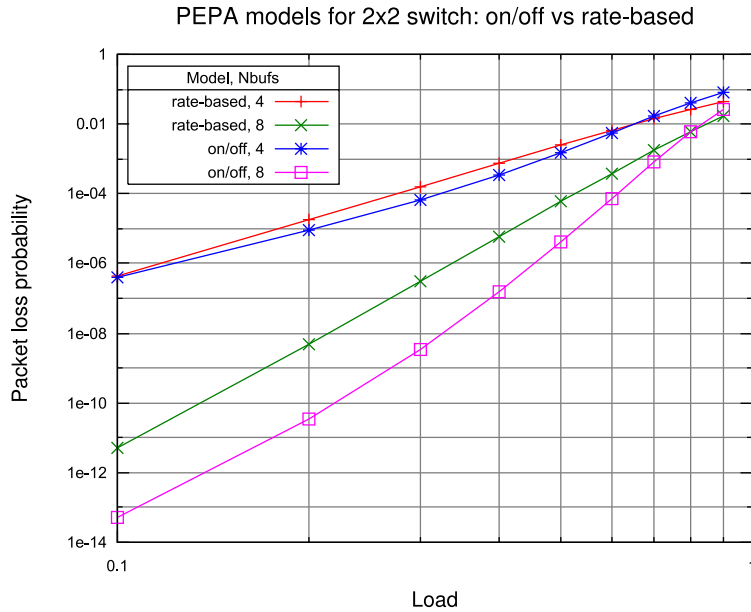

Fig. 8. Comparison of rate-based and two-state models

V. CONCLUSION

We have presented a methodology for analytical modelling of steady-state packet loss in an asynchronous packet switch. The model uses a two-state traffic distribution and keeps an explicit count of the buffer occupancy. Packet loss is modeled by introducing "drop" states. The occupancy of the drop states is a measure for the packet loss. The approach allows to model complex switches without having to resort to discrete-event simulations. However, the state space can very large, and building a PEPA model with explicit states is non-trivial. The results from the PEPA model agree well with the results obtained from discrete event simulations.

REFERENCES

[1] W. Vanderbauwhede, D. Harle, and D. G. Smith, "OPSnet: an optical packet switched network," in Proc. PHOTON02, Cardiff, UK, Sept. 2002.
[2] W. J. Dally and B. Towles, "Route packets, not wires: On-chip interconnection networks," in *Proc. DAC*, Las Vegas, NV, USA, June 2001, pp. 684–689.
[3] W. Vanderbauwhede, "Generic template-driven simulation automation tool," CPAN, Aug. 2004. [Online].
Available: http://search.cpan.org/author/WVDB/Simulation-Automate-1.0.1