Parallel pipeline to ATM: Graphical simulation techniques

M. Sulieman, C. H. Ooi, and M. Fleury*

Abstract

Experience gained from constructing a graphical simulator of pipelines of processor farms has been transferred to prototype simulations of an ATM buffered-Banyan switch, and a slotted-ring network. Ways to make a meaningful and appealing display are indicated. The qualitative information that can be usefully shown is considered. The design and evaluation of the simulators is included.

1 Introduction

The development of a graphical simulation tool for pipelines of processor farms (PPFs) [1] has led us to consider whether similar graphical techniques could be transferred to the simulation of Asynchronous Transfer Mode (ATM) switches and networks. ATM switches broadly share with PPFs a concern for latency, throughput, and buffering to smooth stochastic flows. The result was two graphical prototypes also written in the programming language Java for portability. One prototype was for examining traffic flows through a single Banyan switch [2] and the other was for a set of switch elements feeding a ring network [3]. The graphical simulations show dynamically changing ATM traffic patterns rather than the summary statistics available from non-graphical methods. Instantaneous quantitative information is also made available to the user. The prototypes were intended to develop appropriate graphical representations and to validate potential uses for a future comprehensive toolkit.

In the parallel computing community, there has long been interest in easing the difficulty of constructing parallel applications by means of graphical tools [4]. Continuous-flow real-time systems with a single flow of data across the pipeline, such as radar, vision, image-processing, and some varieties of speech recognition, are potential applications of the PPF variety of parallelism. Each stage of a parallel pipeline can be a processor farm [5], in which all global operations are performed at the farmer or data-manager. In general, pipelines with synchronous data flows [6, 7, 8] can be conveniently modelled through order statistics [9]. However, the intention of the PPF simulator was to model *asynchronous* data flows to establish whether the resultant pipeline traversal latency and throughput met specification targets. Some progress has been made in combining delay distributions for buffer queues with extremal statistics for

*The authors are with Dept. of Electronic Systems Engineering, University of Essex, Wivenhoe Park, Colchester, CO4 3SQ, {mhsuli,chooi,fleum}@essex.ac.uk

service rates [10] but results are limited to exponential distributions. Therefore, graphical simulation seemed apt.

Graphical simulation is potentially a 'simple and intuitive' way to make ATM network design accessible. The variety of switch designs [11] and potential traffic patterns makes a graphical simulator a way . ATM systems are by nature suited to discrete-time simulation as cells arrive synchronously at a switch. The effect of differing cell rejection policies, taking into account cell priority, can be included. Delay jitter (coefficient of variation) [12], can be found recursively. Analytic results are available for switch elements [13], but are difficult to simulate as the low loss rates require lengthy simulations at the cell scale. The VISTA graphical simulator [14] for a tandem Banyan switch employed a cell-loss matrix representing the loss rate at each switch element by shade of colour, though the value of this representation is not established unless the simulator is used for didactic purposes. To run simulations to detect cell loss probabilities of the range 10^{-8} to 10^{-12} requires burst scale simulation [15] or aggregated simulation [16], which is event-driven. Parallel simulation with rollback was ruled because of a desire to make the simulation accessible to those with a PC or laptop.

2 General display design principles

From our experience with the parallel pipeline simulator we have found, at the risk of being over prescriptive, the following general points important to grasp. The point of employing graphical means is to convey to the user meaningful information in a qualitative way. Access to quantitative information may then reinforce that meaning. Any tool should be be designed so that it presents no significant obstacles to its use, particularly when first encountering the tool. Failure to observe these principles may result in a display that is superficially attractive but is not helpful to users.

On the premise that the user will want from a simulation a broad picture of the behaviour of the phenomenon we have represented running means, whether it is mean job flow along the pipeline or mean cell flow across an ATM switch. Movement of individual jobs or cells is faithful to actuality but cannot easily be integrated by the user. While motion of individual packets can produce screen flicker despite facilities such as `MediaTracker` within the Java development toolkit (JDK), colour can be quickly changed on a screen. Gradated colour tones are acceptable to those with impaired colour vision. Ideally, the colour range should have a meaning able to be verbalised. For example, in the pipeline simulator the colours 'heat up' as the traffic flow intensifies. An additional representation used by us is to alter the width of lines as traffic flows change. Change within the system of interest unfortunately does not always take place at the rate one would like. Therefore, we have found it is still necessary to mark progress. In the parallel pipeline, the passage of a job packet along a link is shown by a arrow opening, and in an ATM prototypes, the arrival of a cell is shown by a change in colour of an input marker to a switch.

Quantitative information is not neglected in the displays but is generally available by clicking on a component. For example, it is important to know from a data farm if one of the worker processors is under-utilised, which can be shown in qualitative terms by changes in the colour of a processor node and in precise terms by clicking on a node for a summary of its activity. Cell loss rate in the ATM simulation may only be usefully represented as a changing number when the traffic is realistic. When the traffic sources are deliberately distorted to illustrate an effect then the ring ATM adapts the cell-loss matrix of VISTA by changing the coluour of the switches.

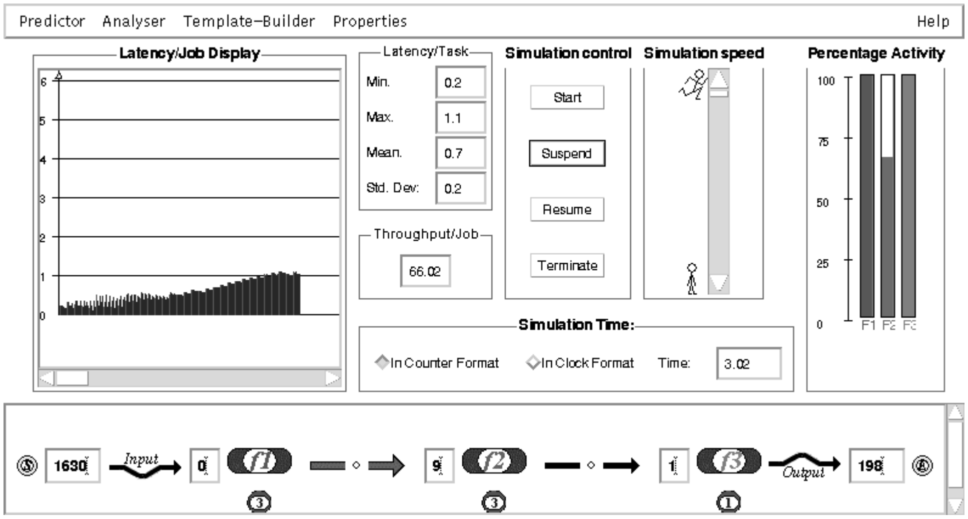
Whatever the validity of the simulation representation, a weak display layout will compromise a design in the user's mind. Weakness can be assessed in terms of symmetry, utilisation of display space, and density of features. The merit or otherwise of a display is not immediately apparent to the designer so that we have found examination by experienced users is beneficial; a first design is rarely adequate. At whatever level, a user is wary of 'learning curves' so we have made an effort to make first use of a tool elementary, in one case adopting the standard 'wizard' format and in all cases adding to the familiarity by means of the standard window layout. The tool designer, being expert, is least qualified to judge how simple to make first use of the tool. Taking as an axiom that the designer will not anticipate the user's requirements, change is built-in by choice of settings at the first level, and by a class structure which adequately encapsulates changes to the specification.

Java has a cleaner object-oriented interface than C++ and now has just-in-time (JIT) compilation easing the problem of slower interpreted simulation speeds reported in [17]. We were able directly to transfer the parallel pipeline simulator from PC to Unix-based workstation and are encouraged by the inclusion of the 'Swing' graphical toolkit in JDK 1.2, which we already use in the ATM ring simulator. However, if the screen is updated by painting in all lines rather than updated on a component basis the update is still slow on versions of the Pentium processor running at 233MHz and below.

3 Parallel pipeline simulation

Critical to running embedded pipelined applications is optimal provision of local and interstage buffering which is dependent on dynamic contingencies. Job service time on any one data farm can be chosen from a variety of distributions making discrete-event simulation appropriate, though also making graphical representation that more difficult because of an inconsistent time representation. Input to the simulator are service time second-order statistics taken from timings made on sequential code prior to parallel decomposition. The distribution of the service times may be estimated by (say) a Kolmogorov-Smirnov test [18]. On the multicomputers of interest, as inter-processor communication times were assumed to be from infinitely divisible distributions mean-value representation was possible. Because soft real-time systems were considered, and because the analysis was at the level of system analysis, not for algorithmic performance tuning, qualitative results were of value. Though in fact, a case study for an automatic handwritten postcode recognition system showed accuracy to within 10% between simulation and a multicomputer running that application [19]. A two-parameter model of parallel computer architecture, endorsed by benchmarking practice, allowed cross-architectural comparisons to be extrapolated from a set of timings on the target machine [20].

A problem with previous analysis visualisation tools for parallel systems, such as ParaGraph [21] written with cumbersome X-lib calls, is that an animated display occurs, rather like a cartoon film. The user may find it difficult to establish a pattern. Moreover, in seeking generality, with twenty-four ways of presenting data, no structure to the ParaGraph's use was provided, for example see comments in [22]. An over-animated display can also reinforce the sequentiality of the simulation whereas in our case the pipeline represented has local and general parallelism.



Latency/Task

Min.	0.2
Max.	1.1
Mean.	0.7
Std. Dev.	0.2

Throughput/Job: 66.02

Simulation Time: ◇ In Counter Format ◇ In Clock Format Time: 3.02

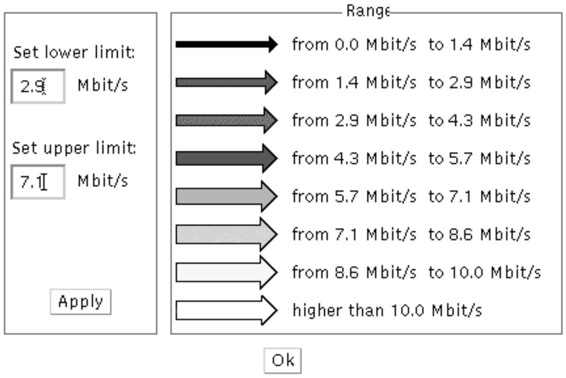
1630 → 0 → 9 → 1 → 198

Figure 1: Main simulation window

Fig. 1 shows a snapshot of the predictor running a postcode simulation. The pipeline backplane occupies the main window with details of the stage activity such as buffer and processor usage available from subsidiary windows. Processor activity is shown using colour by analogy with stop/go displays. Again using the linguistic associations of colour, the communication arrows change colour from black, through red to white to highlight 'hotspots'. The arrows also widen and contract. However, the cumulative mean bandwidth, not instantaneous bandwidth is displayed. The colour scaling is adjustable to centre on critical data rates as otherwise the variation across the whole bandwidth range is too low to show up, Fig 2. Latency is also indicated in a persistent display. Jobs can be grouped into tasks to arrive at an optimal regime [7]. Jobs are marked off at task boundaries, with the task latency determined by the slowest job. Though persistent displays convey more information, they need to be balanced with features marking progress, which is why the processor activity diagram and message motion arrows are included.

4 ATM architecture

Traffic sources for the two ATM simulators are represented by a burst/silence model [23]. Common choices of distribution particularly for video, are Poisson and Bernoulli streams.



Set lower limit: 2.9 Mbit/s

Set upper limit: 7.1 Mbit/s

Range

from 0.0 Mbit/s to 1.4 Mbit/s

from 1.4 Mbit/s to 2.9 Mbit/s

from 2.9 Mbit/s to 4.3 Mbit/s

from 4.3 Mbit/s to 5.7 Mbit/s

from 5.7 Mbit/s to 7.1 Mbit/s

from 7.1 Mbit/s to 8.6 Mbit/s

from 8.6 Mbit/s to 10.0 Mbit/s

higher than 10.0 Mbit/s

Figure 2: Scaling window

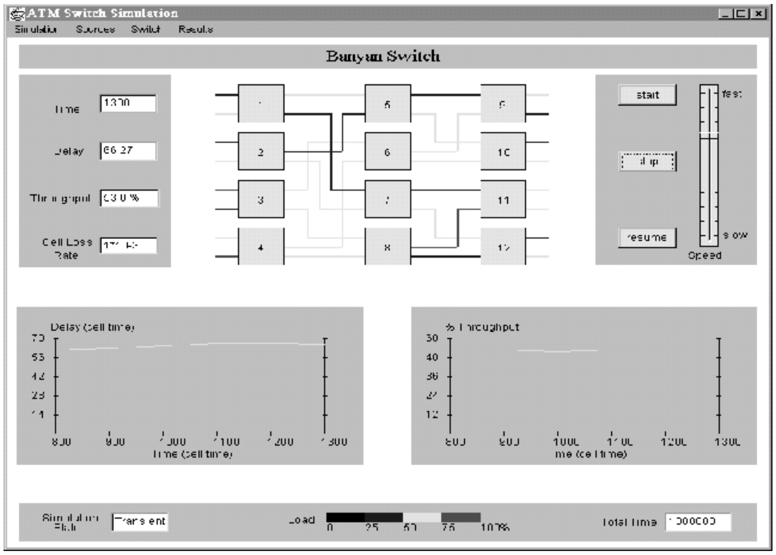
4.1 Multistage switch

The Banyan multi-stage switch, with input buffering at each switching element, has the advantage that the traffic load is distributed and no routing control signals are needed. Without a prior sorting network, throughput can be low if FIFO queueing at buffers is employed [24], with commensurate cell losses. [25] is an elegant queueing model for the Banyan switch, which, without contention resolution through prior sorting, is shown to service traffic intensities of about 0.45.

Input buffering is normally subject to head-of-line (h.o.l.) blocking which can be alleviated by a non-FIFO RAM buffer. Alternatively, for a 2×2 switching element, output buffering or buffering at cross-points combined with internal speedup is possible [26]. It is these permutations which would appear to make a simulation more suitable for practical usage.

The main window of the switch simulator which is symmetrically arranged with little inactive space is shown in Fig. 3. The central feature of the user interface is the switch whose internal and external links change in colour according to the different traffic patterns and load values. Lack of display space prevented changing the widths of links. The status panel at the bottom gives permanent and semi-permanent information about the simulation. Importantly, there is user control of simulation speed and re-runs. Summary statistics in qualitative and quantitative form are included in the front panel. The graph windows scroll, conveying time-wise variation in delay and throughput. The square components representing the switching elements with two input crossbar, can be clicked to give an animated image of that element, Fig. 4. By setting the relative speed of the switching elements *vis-a-vis* the external link speed, as well as the buffer size, the user can greatly effect the behaviour of the switch.

ATM switches can be driven by a back-pressure algorithm [27], and this algorithm has also been employed to drive the update order of the simulation after each timing round. In the back-pressure algorithm, a cell is only transmitted if there is forward buffer space available, which applies recursively across the switch stages. In the ATLAS 1 i.c., multi-lane back-pressure is employed to



Time 1300

Delay 66.27

Throughput 43.0 %

Cell Loss Rate [illegible]

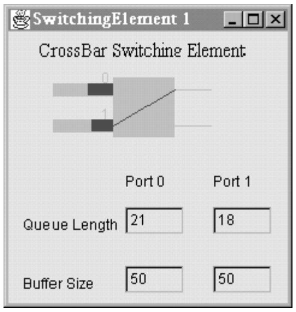
Simulation Stat. Transient

Total time 1000000

Figure 3: Main window of switch simulator

counter head-of-line blocking.

In the case of a collision at the output of a switch element, random arbitration logic is simulated. Other known choices are: cyclic; state-dependent; and delay dependent [28], which again would be necessary in a comprehensive tool.



	Port 0	Port 1
Queue Length	21	18
Buffer Size	50	50

Figure 4: Switching element display

The simulation was verified with runs of 500,000 time slots against known analytic and simulated results [25, 26] for throughput, Fig. 5. Delay and cell loss rate are also included in the figure.

4.2 Ring network

A 156 Mbps slotted fiber-optic ring, single or with dual counter-rotating rings[1], has been proposed [29] and tested using a variant of the Orwell slotted ring

[1] Note that the capacity of a spatial re-use slotted dual ring with shortest path routing is four times that of a single ring.

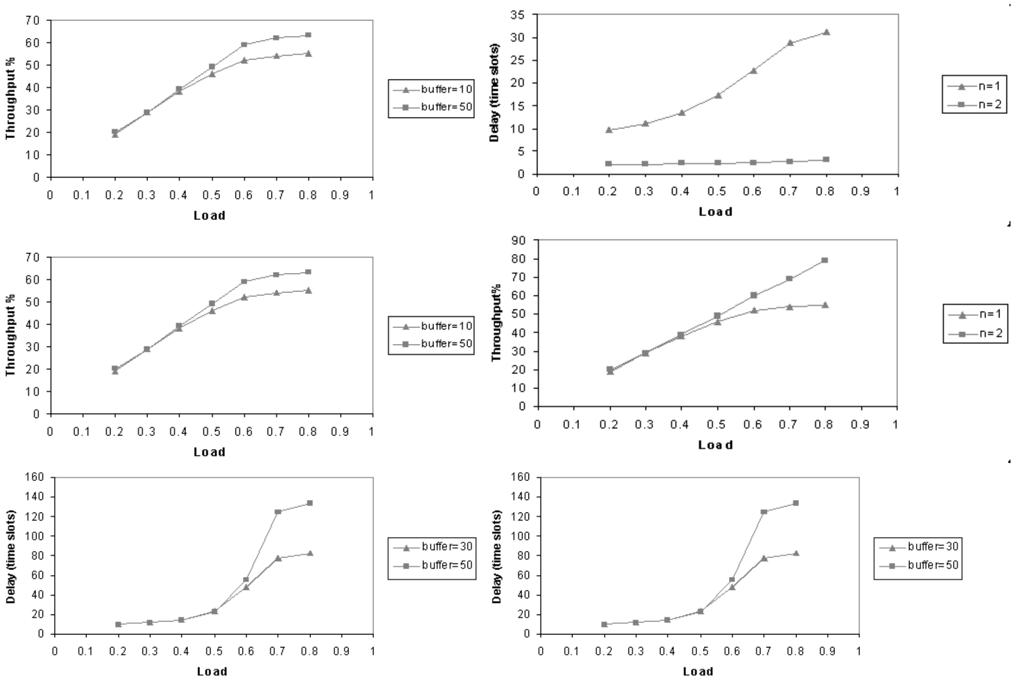


Figure 5: Banyan evaluation

protocol for ATM LANs and subsequently modelled in [30]. In our graphical simulator, an ATM ring (ATMR) is modelled with a single transmission queue per station, each generating traffic at rates up to 10 Mbps, with estimates of ring transfer delay found.

Again, diversity in design exists, principally in the form of an access protocol to provide fairness but avoid blocking. ATMR stations during each fairness cycle fill any available slots until its allocated window has been used up. The 'Check Quota before Transmission' (CQBT) protocol requires a snoopy mechanism, examining the upstream station's traffic to determine local congestion before . Fair and Efficient Cyclic Control Algorithm' (FECCA) [31] includes a global corrective by means of signalling via the counter-rotating ring.

The display window of the ring network is shown in Fig. 6 for a single ring. The ATM station links change when traffic is present and the station is highlighted when cell loss occurs. The central ring grows in width with increased global traffic. For the prototype, a single ring system was simulated. JDK 1.2 card index widget allows graphs of queueing delay and throughput on a per-station basis to be cycled through. Instantaneous information on individual stations is still popped up. The effect of setting the priority bit on cell loss for either push-out or partial schemes [32] can be judged in the simulation. In fact, there are a variety of schemes to relieve congestion such as pre-emptive cell discard to avoid discarding a run of cells, disconcerting to the viewer of video sequences. We found it important to provide facilities for the user to load and save simulations, find confidence intervals, and in general to manage the simulation results. Comparisons with the performance and behaviour of other switches by running simulations side-by side is possible.

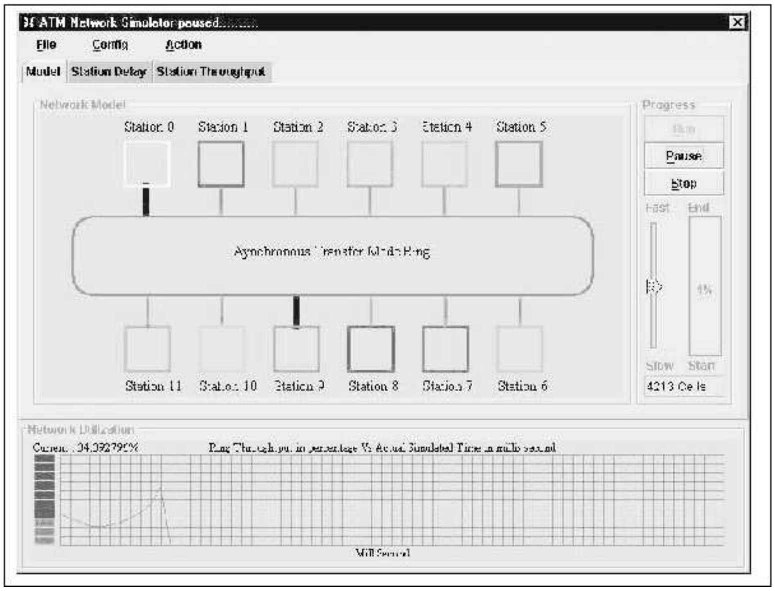


Figure 6: Main window of ring network simulator

Window size during a fairness cycle is the critical factor in ring utilisation. If traffic is homogeneous (uniform distribution sources), the reset cycle degrades performance over a pure slotted ring [33]. The simulator was validated with a burst (Poisson distribution with mean 50 cells) – silence (negative exponential with mean 1000 cells) traffic source model (single source per station – 12 stations). The destinations were randomly chosen with shortest path routing. Buffering was normal. The ring length was set at 50km with a medium propagation delay of 5μs/km. Fig. 7 shows a decrease in delay for all stations with increasing window size. Station throughput transmitted cells/available slots, declines if the total number of cells transmitted by each station is fixed. In fact, the opportunity to transmit increases with window size, as the available slots increases. Thus, network throughput increases, as Fig. 7 shows. Note that the number of reset periods are also reduced with increased window size.

4.3 Design process

Since the facilities of such tools will almost certainly be extended or modified a considered (object-oriented) design process was needed, going from a high-level design to more detailed consideration, Fig. 8. In general, an object-oriented approach is well-suited to a hierarchical construction of ATM simulations from a database of components: switches, clocks, ports, controllers. The Libra system[34], based on the well-known Ptolemy system, is a way of putting together single-model simulations of an ATM LAN in this fashion, though not as a graphical simulation.

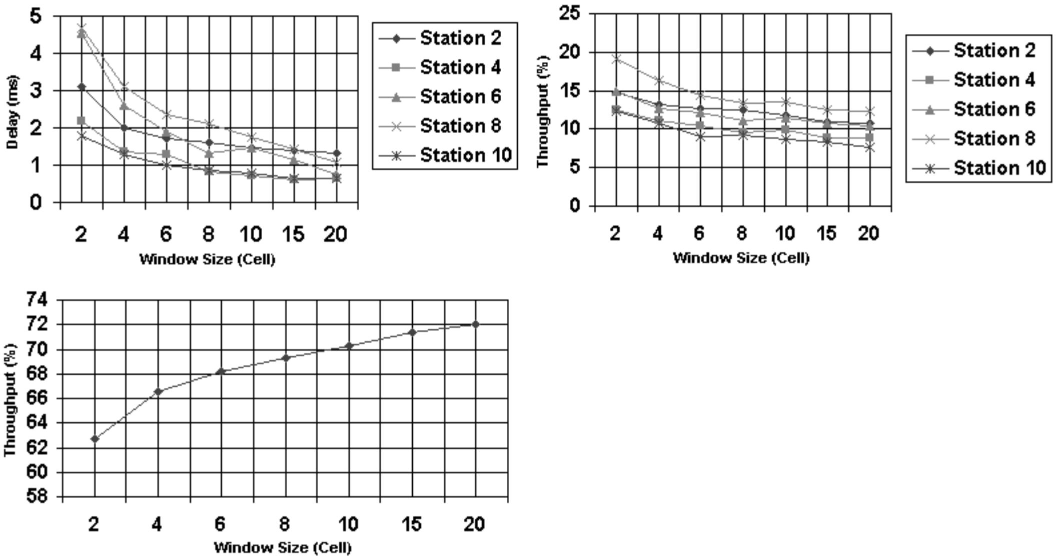


Figure 7: ATMR evaluation

5 Discussion

Though updating the screen in a regular manner through discrete-time (cell scale) simulation was appealing it was felt that event-driven simulation (burst scale) would be preferable for future implementations at a small loss in accuracy. To record each timing for the evaluation runs of Section 4.1 took 12 mins. on a Pentium 266MHz processor. However, bear in mind that general usage would be short runs trying out the effects of various changes without accuracy in mind. The speed of display was also noticeably enhanced on the 450MHz processors of the Pentium family already available. Scalability of display as more stations or switches are added is an issue which would probably be addressed by zooming in, thus keeping the number of screen updates constant. However, current non-graphical switch simulations do not apparently stray beyond 128 elements. Despite the more detailed graphics employed in the ring simulator the advantage is lost if these cannot be updated in component fashion.

6 Conclusions

We have assembled a set of graphical techniques which convey meaning to the user, helping to firm up network design decisions. Colour, carefully-considered layout, differentiation between the types of information presented, as well as user extensibility are needed. Graphical simulation of parallel pipelines at the system level is a well-posed problem. When simulating ATM switches in a graphical manner, careful consideration is needed as to what is required of the simulation. Traffic flow patterns, mean latency, mean throughput, and scenarios with comparatively high cell-loss probabilities are well represented, very low cell-loss probabilities are not so easy to represent. A further difficulty, which in principle can be resolved by systematic construction of a database, is the variety in switch designs, buffering arrangements (input, output, and recirculation), queueing policy, input policies, buffer rejection policies and so on. This is not such a daunting prospect as once a structure is available the details can be included as and when they are needed. The advantage of a tool of this sort is that it can consolidate knowledge, presenting a bewildering array of alternatives in a structured and convenient format. More detail enhances the credibility of the simulation, whereas analytic techniques give 'broad brush' results. However, a graphical simulator is more suited to answering 'what if' questions by an intranet planner or as a demonstrator. There is an obvious didactic use as well. Switch design will always require more specificity than a graphical presentation can give.

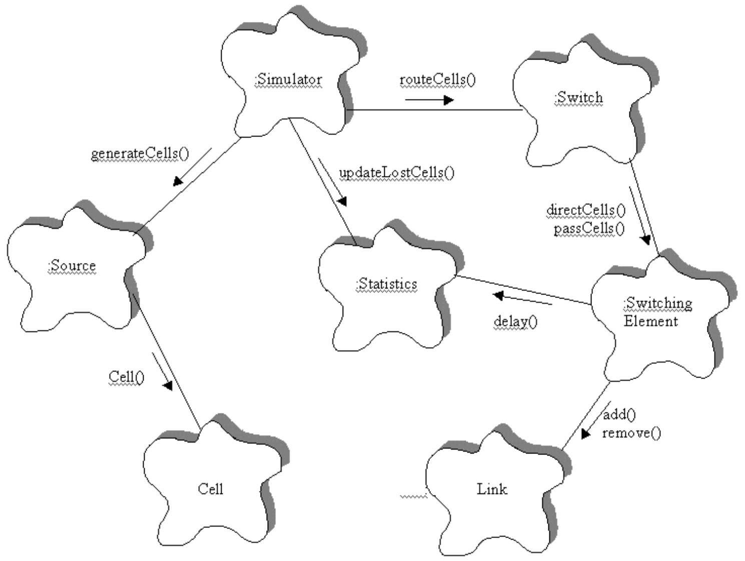


Figure 8: Booch-style diagram for the switch simulator

References

[1] A. Çuhadar, A. Downton, and M. Fleury. A structured parallel design for embedded vision systems: A case study. *Microprocessors and Microsystems*, 21:131–141, 1997.

[2] S. Cheemalavagu and M. Malek. Analysis and simulation of Banyan interconnection networks with 2 × 2, 4 × 4 and 8 × 8 switching elements. In *Real-time Systems Symposium*, pages 82–89, 1982.

[3] J. L. Adams and R. M. Falconer. Orwell: A protocol for carrying integrated services on a digital communications ring. *Electronic Letters*, 20(23):970–971, 1984.

[4] C. Pancake. Visualization techniques for parallel debugging and performance tuning tools. In A. Y. Zomaya, editor, *Parallel Computing: Paradigms and Applications*, pages 376–393. Thomson, London, 1996.

[5] A. S. Wagner and S. T. Sreekantaswamy, Chanson. Performance models for the processor farm paradigm. *IEEE Transactions on Parallel and Distributed Systems*, 8(5):475–489, May 1997.

[6] M. Dubois and F. A. Briggs. Performance of synchronized iterative processes in multiprocessor systems. *IEEE Transactions on Software Engineering*, 8(4):419–431, July 1988.

[7] C. P. Kruskal and A. Weiss. Allocating independent subtasks on parallel processors. *IEEE Transactions on Software Engineering*, 11(10):1001–1016, October 1985.

[8] S. Madala and J. B. Sinclair. Performance of synchronous parallel algorithms with regular structures. *IEEE Transactions on Parallel and Distributed Systems*, 2(1):105–116, January 1991.

[9] N. Balakrishnan and A. C. Cohen. *Order Statistics and Inference*. Academic Press, Boston, 1991.

[10] M. Fleury, A. C. Downton, and A. F. Clark. Modelling pipelines for embedded parallel processor system design. *Electronic Letters*, 33(22):1852–1853, 1997.

[11] E. W. Zegura. Architectures for ATM switching systems. *IEEE Communications Magazine*, pages 28–37, February 1993.

[12] R. Steinmetz. Human perception of jitter and media synchronization. *IEEE Journal on Selected Areas in Communications*, 14(1):61–72, January 1996.

[13] S. Monaghan. Exact analysis in discrete time of 2×2 ATM switch serving correlated traffic at both inputs. *Electronic Letters*, 33(18):1528–1529, 1997.

[14] N. Tummala and K. A. Robbins. VISTA: a configurable visualization and simulation tool for ATM switches. In *MASCOTS'96*, pages 223–226, 1996.

[15] J. M. Pitts. Cell-rate modelling for accelerated simulation of ATM at the burst level. *IEE Proceedings Communications*, 146(6):379–385, 1995.

[16] F. Hao, I. Nikolaidis, and E. W. Zegura. Efficient simulation of ATM networks with accurate end-to-end statistics. In *International Conference on Communications*, 1998. Available from `http://www.cc.gatech.edu/people/home/fang`.

[17] R. McNab and F. W. Howell. Using Java for discrete event simulation. In J. Hillston and Pooley R., editors, *UK PEW 1996*. University of Edinburgh, 1996.

[18] D. E. Knuth. *The Art of Computer Programming*, volume 2/ Seminumerical Algorithms. Addison-Wesley, Reading, MA, 2 edition, 1981.

[19] M. Fleury, N. Sarvan, A. C. Downton, and A. F. Clark. Methodology and tools for system analysis of parallel pipelines. *Concurrency: Practice and Experience*, 1999. Accepted for publication.

[20] N. Sarvan, R. Durrant, M. Fleury, A. C. Downton, and A. F. Clark. Analysis prediction toolkit(apt) for real-time image processing. In *IEE Image Processing and its Applications, IPA99*, 1999. Accepted for publication.

[21] M. T. Heath and J. A. Etheridge. Visualizing the performance of parallel programs. *IEEE Software*, 8(5):29–39, 1991.

[22] I. Glendinning, V. S. Getov, S. A. Hellberg, and R. W. Hockney. Performance visualisation in a portable parallel programming environment. In G. Haring and G. Kotsis, editors, *Performance Measurement and Visualization of Parallel Systems*, pages 251–275. Elsevier Science, Amsterdam, 1993.

[23] E. R. Rathgeb. Modeling and performance comparison of policing mechanisms for ATM networks. *IEEE Journal on Selected Areas in Communications*, 9(3):325–334, April 1991.

[24] Y. Shobatake and T. Kodama. A cell switching algorithm for the buffered banyan network. In *International Conference on Communications*, volume II, pages 754–760, 1990.

[25] Y-C. Jencq. Performance analysis of a packet switch based on single-buffered banyan network. *IEEE Journal on Selected Areas in Communications*, 1(6):1014–1021, December 1983.

[26] E. R. Rathegeb, T. H. Thelmer, and M. N. Huber. Buffering concepts for ATM switching networks. In *Globecom'88*, volume III, pages 1277–1281, 1988.

[27] G. Kornaros, D. Pnevmatikatos, P. Vatsolaki, G. Kalokerinos, C. Xanthaki, S. Mavroidis, D. Serpanos, and M. Katevenis. ATLAS 1: Implementing a single-chip ATM switch with backpressure. *IEEE Micro*, 19(1):30–41, Jan/Feb 1999.

[28] R. Händel, M. N. Huber, and S. Schröder. *ATM Networks: Concepts, Protocols, Applications.* Addison-Wesley, Harlow, UK, 1998.

[29] K. Imai, T. Honda, H. Kasahara, and T. Ito. ATMR: Ring architecture for broadband networks. In *Globecom'90*, volume 3, pages 1734–1738, 1990.

[30] C. Bach and S. Richter. Performance evaluation of an ATM-based high-speed slotted ring network. In *Transport Technology for Broadband Optical Access Networks*, pages 259–270, 1993. SPIE volume 1974.

[31] I. Rubin and H-T. Wu. FECCA — a new access algorithm for an ATM ring network with destination removal. In *IEEE INFOCOM*, volume I, pages 368–374, 1993.

[32] V. F. Hartano and H. R. Sirisena. User-network policer: A new approach for ATM congestion control. In *IEEE INFOCOM*, volume I, pages 376–383, 1993.

[33] C. Bach and S. Richter. Major performance characteristics of the ATMR MAC protocol in the Gbit/s range. In *Broadband Network Strategies and Technologies*, pages 338–350, 1995. SPIE volume 2450.

[34] W-S. Chen, C-H. Lin, and Y-C. Chen. Design and implementation of an object-oriented ATM network simulator. In *MASCOTS'96*, pages 123–127, 1996.