2010

A new integrated unicast/multicast scheduler for input-queued switches

Kwan-Wu Chin
University of Wollongong, kwanwu@uow.edu.au

A new integrated unicast/multicast scheduler for input-queued switches

Abstract

Researchers have thus far considered scheduling unicast and multicast traffic separately, and have paid little attention to integrated schedulers. To this end, we present a new integrated scheduler that considers both unicast and multicast traffic simultaneously and also addresses key shortcomings of existing approaches. Specifically, we outline a scheduler that achieves 100% throughput, and unlike existing schemes, do not require a tuning knob. Moreover from our extensive simulation studies, we show that it works well in uniform, non-uniform and bursty traffic scenarios.

Disciplines

Physical Sciences and Mathematics

Publication Details

K. Chin, "A new integrated unicast/multicast scheduler for input-queued switches," in The 8th Australasian Symposium on Parallel and Distributed Computing (AusPDC'10), 2010, pp. 13-20.

A New Integrated Unicast/Multicast Scheduler for Input-Queued Switches

Kwan-Wu Chin
School of Electrical, Computer and Telecommunications Engineering
University of Wollongong
kwanwu@uow.edu.au

Abstract

Researchers have thus far considered scheduling unicast and multicast traffic separately, and have paid little attention to integrated schedulers. To this end, we present a new integrated scheduler that considers both unicast and multicast traffic simultaneously and also addresses key shortcomings of existing approaches. Specifically, we outline a scheduler that achieves 100% throughput, and unlike existing schemes, do not require a tuning knob. Moreover, from our extensive simulation studies, we show that it works well in uniform, non-uniform and bursty traffic scenarios.

1 Introduction

The Internet is growing at a rapid pace, driven by the proliferation of high bandwidth applications capable of delivering voice and video traffic. This is particularly evident on Internet 2, where such applications are being used to deliver television programs, lectures, conduct video conferences, and to create interactive and collaborative research environments [1]. As a result, given their high bandwidth demands, Internet service providers are in need of switches/routers that are capable of switching unicast and multicast cells at high speeds.

To date, researchers have proposed a myriad of router designs capable of switching packets or cells at speeds ranging from gigabits to terabits per-second; see [5]. The most popular design is based on the input queued architecture, as it has good scalability with respect to switch size and link rate [5]. Figure 1 shows a block diagram of one such router with N inputs and N outputs connected by a crossbar fabric. It operates in cell mode where variable length packets are fragmented into fixed size cells before traversing the crossbar. They are then re-assembled at their respective output before leaving the router [8]. Each input has N virtual output queues (VoQs) for storing the corresponding unicast cells of N outputs. Without VoQs, a router will experience the head of line (HOL) blocking problem, which limits its throughput to only 58.6% [4]. Unlike previous router designs [3][15][6], which maintain $k < 2^N - 1$ multicast queues, our router has a single multicast queue and N staging buffers; their use will be explained in Section 3.

The scheduler is a key component of any high speed routers. It is responsible for arbitrating cells/packets from input ports across a switching fabric to output ports. Ideally, the scheduler must have 100% throughput and low complexity. In this respect, a significant amount of work has been devoted to unicast scheduling algorithms, the most popular being iSLIP [10]. Similarly, a lot of efforts have been devoted to developing high speed multicast scheduling algorithms. Examples include Concentrate, TATRA and WBA [12], ESLIP [9] and Max-Scalar [6]. However, little attention has been paid to integrated schedulers. That is, a scheduler that considers both unicast and multicast cells simultaneously rather than separately.

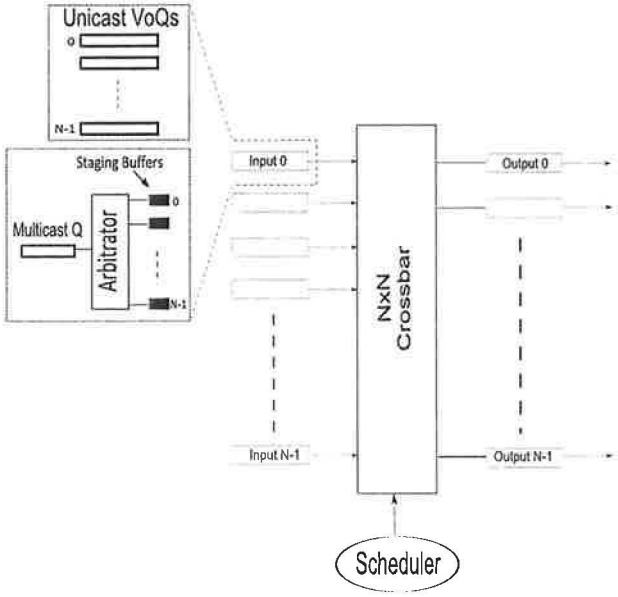

Figure 1: Input-queued switch architecture. Also shown is an *Arbitrator* managing N staging buffers.

This paper, therefore, adds to the existing state-of-the-art by proposing an integrated scheduler that overcomes limitations with existing approaches. Specifically, it works in conjunction with staging buffers to overcome the multicast cell HOL problem. Moreover, the proposed scheduler considers the weight of both unicast and multicast cells simultaneously, and hence works well in both uniform and non-uniform traffic scenarios. Our simulation studies involving uniform, non-uniform and bursty traffic sources show that our scheduler has 100% throughput, fair to both unicast and multicast traffic, and achieves superior performance over existing schedulers. Lastly, unlike Zhu et al. [16]'s scheduler, our scheme does not involve a tuning knob. This is a significant advantage because it frees the scheduler from continuously adjusting its behavior with changing traffic conditions.

This paper is organized as follows. We first review existing works and highlight their limitations in Section 2. After that, Section 3 outlines our integrated scheduler and the aforementioned staging buffers. Then, in Section 4, we discuss our simulation parameters. Section 5 presents our experimentation results on a NxN switch over varying traffic load and cell types. We then discuss our results in

Section 6, before concluding in Section 7. Note, in our discussions to follow, we use the term router and switch interchangeably.

2 Background

Each multicast cell has a fanout set that specifies its outgoing outputs. This is the key reason that complicates multicast cells scheduling, especially when cells have varying fanout sizes that can range from 1 to N, assuming a NxN switch. Hence, in each time slot,a router's load can increase by N^2. Moreover, Andrews et al. [2] have shown that scheduling multicast cells is a NP-hard problem. Besides that, there is also the HOL multicast cell blocking problem. Assume cell C_1 and C_2's fanout vector is {0,1,2,3} and {1,2} respectively. If C_2 is queued behind C_1, then it will have to wait until all destination outputs of C_1 have a received a copy of C_1 before it receives service; i.e., at least four time slots. Note that each cell will have to contend with other multicast and unicast cells headed to the same output. Clearly, a switch's performance degrades when it persistently receives multicast cells with a large fanout. One naive approach to address this problem is to have $2^N - 1$ queues, where each queue stores cells headed to the same set of outputs. Unfortunately, this solution is not scalable, especially in large switches. Hence, researchers, such as [3], use k multicast queues instead, where $k < 2^N - 1$. As a rule of thumb, for a switch with N outputs, 2N multicast queues are needed to ensure good performance. We will show in Section 3 how staging buffers reduce this memory requirement further by storing only the address of a multicast cell.

As mentioned earlier, most researchers have developed schedulers that are optimized for either unicast or multicast traffic, and not many are designed for both unicast and multicast cells. In fact, only a handful of schedulers exist. Andrews et al. [2] propose that inputs transmit unicast traffic to outputs left unmatched by the multicast scheduler. Unfortunately, their approach leads to the starvation of unicast flows, and does not address the HOL blocking problem. Apart from that, their scheme is unfair to unicast traffic because it gives higher priority to multicast cells.

Schiattarella et al. [13] propose an approach that first uses a unicast and a multicast scheduler to independently derive the maximal matchings for unicast and multicast cells. A module then filters and integrates the matchings found from both schedulers in a fair manner. To avoid starvation, the module ensures edges that missed out in the current time slot will receive service in the next time slot. Their approach, however, is unnecessarily complex and do not consider the weight of unicast and multicast cells simultaneously.

Minkenberg [11] proposes to duplicate the address of a multicast cell into VOQs that correspond to its fanout. In effect, treating a multicast session with a fanout size of n as n unicast sessions. They showed that their scheme is better than the Concentrate scheme [12], but unfortunately its performance is worst than Concentrate for input queued switches. In particular, it does not take advantage of a crossbar switch innate multicast ability. Apart from that, it is not scalable, as input buffers need to have high write bandwidth.

In [8], McKeown presents ESLIP, a multicast extension of iSLIP [10]. Each input has a multicast queue, and a global multicast pointer a_M that points to the input receiving multicast service. The pointer a_M is updated in a round robin manner after the scheduler has sent a copy of a cell to all outputs in its fanout. In each round, inputs send a request to outputs corresponding to non empty queues. Outputs then consider these requests and send a grant to the input with the highest priority traffic. If that happens to be a multicast cell, the output sends its grant to the input a_M is pointing at. Inputs then send an accept to the output corresponding to its highest priority traffic. ESLIP, however, suffers from the HOL blocking problem, and does not allow different multicast queues to receive service in the same round. Moreover, like iSLIP, it does not perform well when traffic are non-uniform.

Lastly, Zhu et al. [16] propose a scheduler, called slot-coupled integration algorithm (SCIA), that preferentially schedules unicast or multicast cells according to a probabilistic parameter called S_m. Specifically, if a time slot is marked as unicast, outputs will first consider unicast requests from inputs before considering multicast requests. Hence, a multicast request is only granted if there are no unicast requests. Similarly, input ports preferentially accept unicast grants. On the other hand, if a time slot is marked as multicast, then inputs and outputs will process multicast requests/grants first. The main limitation with Zhu et al.'s work is that their approach does not consider the weight of unicast and multicast cells simultaneously. For example, in a multicast time slot, some outputs in a multicast cell's fan-out vector may have higher weighted unicast cells awaiting transmission. Lastly, their scheme is designed for uniform traffic only, and is sensitive to the parameter S_m; as we will show in Section 5.

3 Integrated Scheduler

To address the aforementioned limitations, we propose to have staging buffers at each input, and an integrated scheduler that makes use of them to schedule both unicast and multicast cells simultaneously. Note, we assume fanout splitting, as this ensures the switch is work conserving and have high throughput [6][2]. Also, the switch operates without any speedup.

3.1 Staging Buffers

Each input, see Figure 1, has N staging buffers corresponding to N outputs; each capable of holding the address of one cell. We refer to a buffer corresponding to output-j at input-i as S_{ij}, where i and j ranges from 0 to $N-1$ for a $N \times N$ router. The aim of these buffers is to prevent the HOL blocking problem without having to maintain $2^N - 1$ multicast queues. All buffers are managed by the arbitrator, which is responsible for scanning the multicast queue and determining the next multicast cell destined for a given output. Specifically, when the arbitrator finds an empty buffer, say S_{ij}, it starts looking for the oldest cell in the multicast queue that is headed to output-j. This ensures cells destined for output-j are not transmitted out-of-order. Once a cell is found, the arbitrator stores the cell's address in S_{ij}.

Figure 2 shows an example staging buffers implementation using Ternary Content Addressable Memory (TCAM) and Random Access Memory (RAM) [14]. When a multicast cell arrives, a tag is created using the cell's fanout bitmap b and its timestamp ts; the former is simply a bitstring of length N that identifies the set of outputs; e.g., 101 corresponds to outputs 1 and 2. The later is the modulo of the cell's arrival time and W; i.e., ts is $log_2(W)$ in size. The resulting tag is then associated with the cell's memory address in the cell buffer RAM. Lastly, ts is added to the corresponding per-output timestamp FIFO queues (POTQ).

The arbitrator is responsible for filling the staging buffers with the address of multicast cells. When a staging buffer S_j is empty, the arbitrator executes the following steps:

1. Set $ts = Dequeue(TS[j])$, where $TS[j]$ refers to the HOL ts value of output j's POTQ.
2. Construct tag (j, ts), and perform a TCAM lookup.
3. Copy the returned cell's address corresponding to (j, ts) into S_j.

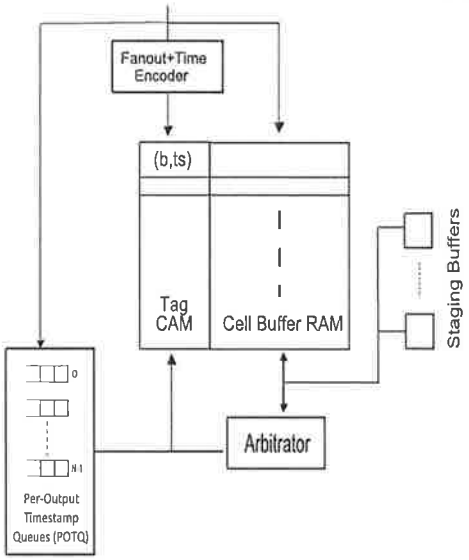

Figure 2: An example staging buffers architecture. b corresponds to the fanout bitmap, and ts is a cell's arrival timestamp (in slot).

After a cell has been transferred, the arbitrator decrements the cell's fanout counter. Here, we assume cells have an associated counter that stores their fanout size. Once the counter reaches zero, the arbitrator frees the memory occupied by the cell.

3.2 Scheduler

Given a $N \times N$ switch with unmatched inputs and outputs, the scheduler executes the following steps at each iteration until no more matches are found:

1. *Request.* Each unmatched input sends a request to every unmatched output corresponding to non-empty VoQs. Requests are also sent for each non-empty staging buffer corresponding to an unmatched output.
2. *Grant.* An unmatched output processes these requests and grants the request with the highest weighted cell. Moreover, the output informs the input whether the grant is due to a unicast or multicast cell.
3. *Accept.* An unmatched input first determines the highest weighted cell with a grant. If it is a unicast cell, the input sends an accept to the corresponding output. However, if the grant is for a multicast cell, the input sends an accept to all outputs that have sent a grant for that cell. In other words, for each staging buffer with a grant and holding the highest weighted cell, an accept is sent to the corresponding output port.

The time complexity at each output is O(2N), since there are N unicast and N multicast requests. In the worst case scenario, the convergence time is O(N) because in each round it is possible only one request is granted. However, in our experiments, convergence time is far smaller than N, especially when multicast cells have a large fanout.

3.3 Example

Figure 3 shows a 3x3 input queued switch. Input-0 has two cells for output 0 and 2, input-1 has a unicast cell for output-0 and also a multicast cell for outputs 1 and 0. Lastly, output-2 has a cell for output-1. In this example, assume that the multicast cells have a higher weight than all unicast cells; this would be the case if they did not receive any service in previous time slots. Moreover, we only show staging buffers at input-1.

Starting at the Request stage, all inputs send a request message to outputs corresponding to non-empty VoQs and staging buffers. Notice that input-1 sends three requests to output-1, corresponding to its unicast and multicast cells. Each output then considers the cells' weight, as specified in each request message, and sends a grant to the input with the highest weight. In this example, input-1 receives two grants from input-0 and 1 respectively, and input-0 has a grant from output-2. Finally, input-0 sends an accept to output-0, and input-1 accepts both output 0 and 1's grant, thereby allowing the cell to be transferred using the crossbar's multicast capability.

4 Simulation Methodology

To study our integrated scheduler, we used SIM [7], and conducted experiments on a $N \times N$ switch; the value of N is specific to the experiment, and the crossbar connecting them has a speedup of one. All simulation runs are for 10 million slots time, and after each run we compute the average latency of unicast and multicast cells. We also record the switch's throughput – the number of matches over the number outputs. All inputs have infinite buffer size. In addition, we use cells' age as weight. Lastly, we set the maximum number of iterations in each time slot to be five.

For comparison purposes, we implemented the following scheduling algorithms:

- *iSLIP-Emulate* [11]. This algorithm creates copies of a multicast cell, and inserts them in VoQs corresponding to the cell's fan-out vector. The VoQs are then scheduled using iSLIP [8].
- *SCIA* [16]. At each input, we maintain $k = 4$ multicast queues, similar to [16]. Cells are always added to the shortest k queue. Apart from that, we set each queue's weight to be the queue length multiply by the HOL cell's age. To determine an appropriate S_m value, we iterate from 0 to 1 at an increment of 0.05 to determine the S_m that provides the best delay to both unicast and multicast cells for a given input load. Lastly, in unicast time slots, we use the oldest cell first (OCF) matching algorithm [8], and for multicast time slots, we use WBA [12]

5 Results

We now present results from our experiments on a NxN switch with uniform, non-uniform and bursty traffic. In addition, we also investigate the impact of different switch sizes.

5.1 Uniform Traffic

Our first experiment is on a 8x8 switch. We generate uniform Bernoulli traffic with a load of 0.1 to 0.55, and designate half of the traffic to be multicast. Lastly, each multicast cell has a random fanout ranging from 1 to 8.

Figures 4 and 5 show the delay incurred by unicast and multicast cells respectively. We see that iSLIP-Emulate, although low in complexity, has the highest unicast and multicast delay. SCIA and our integrated scheduler have comparable unicast and multicast delays. Note that, ours has the advantage of not requiring a tuning knob. In other words, SCIA's performance is achieved by tuning S_m iteratively. As the input load increases, SCIA's multicast delay becomes significantly higher, whereas unicast cells experience lower delays than those scheduled by our integrated scheduler. This is because SCIA has to probabilistically provide time slots to service unicast cells, which

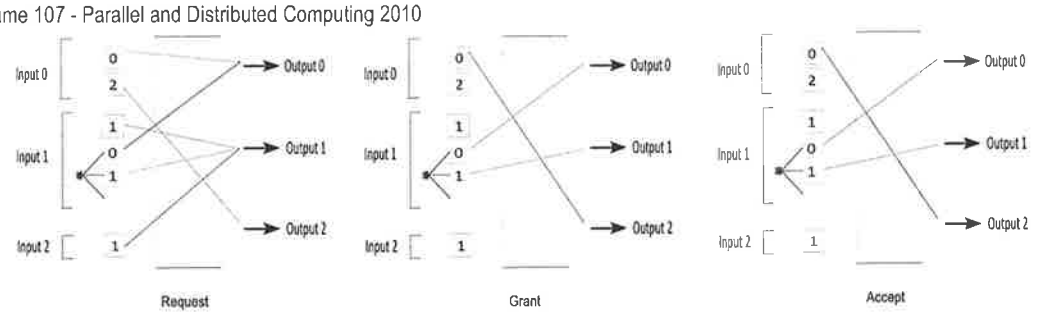

Figure 3: Unicast and multicast scheduling example.

reduces the throughput of multicast traffic, hence increasing delay significantly.

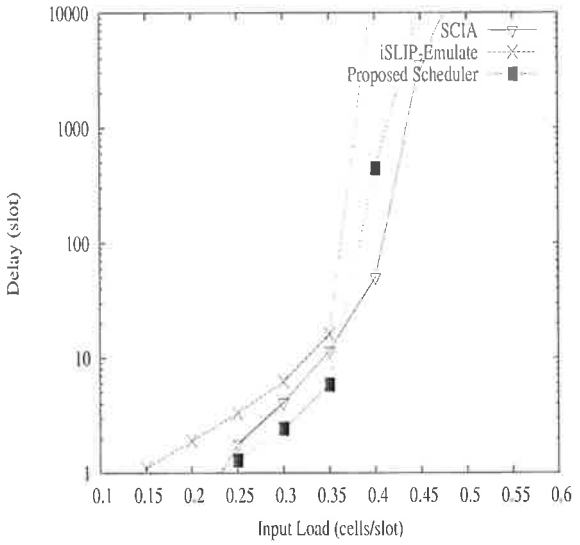

Figure 4: Average delay of unicast cells. Results are for an 8x8 switch with uniform i.i.d Bernoulli arrivals.

On the other hand, our integrated scheduler treats both unicast and multicast cells equally, which results in both traffic types experiencing similar delays. Apart from that, our scheduler utilizes the crossbar fabric's innate multicast ability when the opportunity arises, thereby increasing throughput. This is particularly critical during high loads, as it delays queue instability.

In the next experiment, we study what happens when inputs have increasing multicast cell arrivals. We fix the input load at 0.45, and increase the percentage of multicast traffic slowly from 10% to 55%. Figures 6 and 7 indicate that iSLIP-Emulate has the worst performance, and our scheduler results in both unicast and multicast cells having similar delays. When the percentage of multicast cells is at 35%, the queues in SCIA become unstable. In other words, SCIA is unable to provide sufficient scheduling opportunities to cells. This is exacerbated by the fact that SCIA probabilistically prefer unicast over multicast cells, and vice-versa. On the other hand, our scheduler ensures that the most urgent cells are transferred in a given time slot, hence it is able to delay queues instability.

5.1.1 Impact of Fanout

An important observation is the impact of multicast cells' fanout. To illustrate the detrimental effects of large fanout, we used a 3x3 switch. Input-0 has a single multicast flow that has a fixed fan-out of three, and an input load of 0.33, thereby yielding an effective load of 1.0. Other inputs have unicast flows that transmit cells uniformly across all outputs. In our experiments, we vary their load from 0.1 to 1

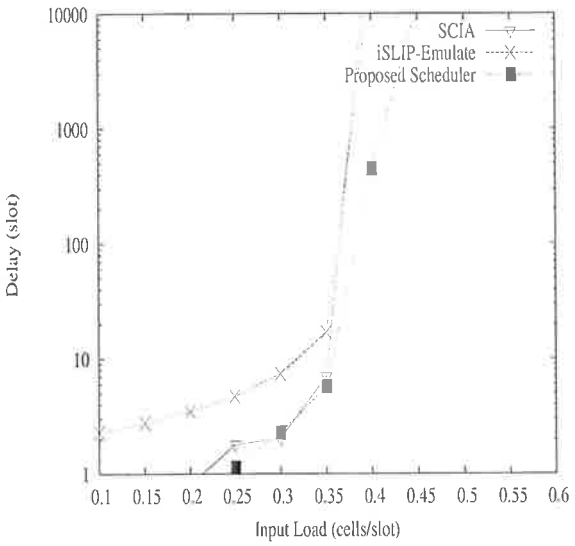

Figure 5: Average delay of multicast cells. Results are for an 8x8 switch with uniform i.i.d Bernoulli arrivals.

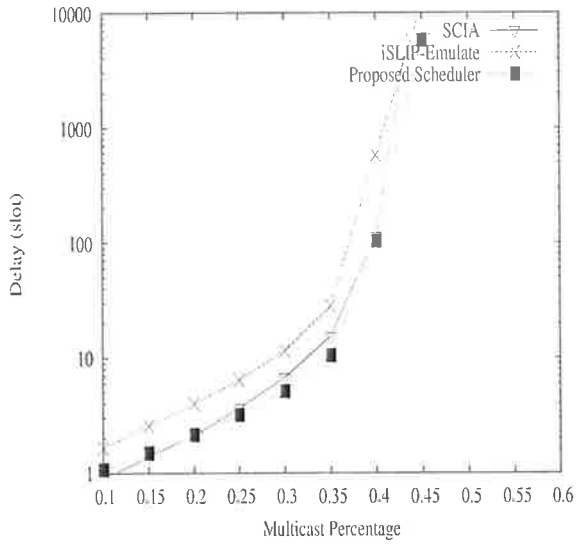

Figure 6: Average delay of unicast cells with uniform i.i.d Bernoulli arrivals.

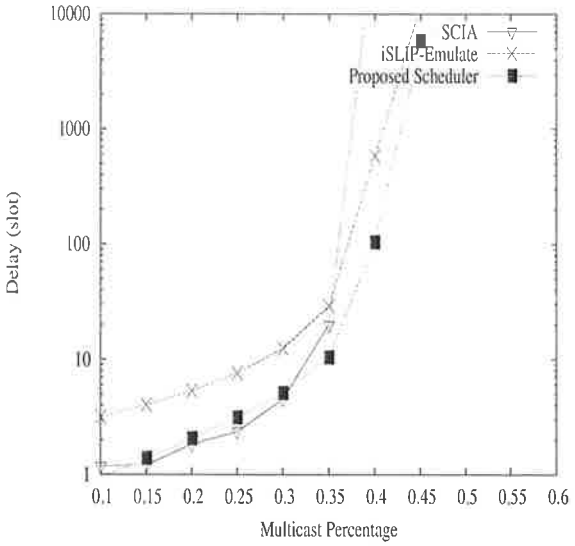

Figure 7: Average delay of multicast cells with uniform i.i.d Bernoulli arrivals.

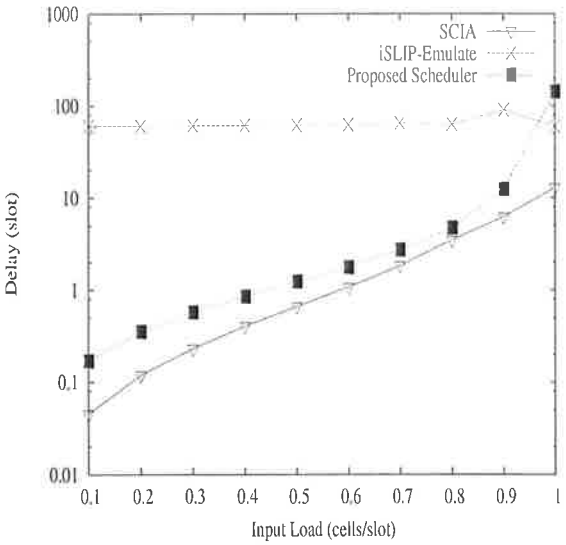

Figure 9: Average delay of multicast cells. Results are for an 3x3 switch with uniform i.i.d Bernoulli arrivals.

to determine their impact on the multicast flow, and vice-versa.

Figures 8 and 9 show the delay incurred by unicast and multicast cells respectively. We see that iSLIP-Emulate has the lowest unicast delay, but has the highest multicast delay. This is due to iSLIPs inability to handle non-uniform traffic, since input-0 has a much higher load than other inputs. SCIA has the lowest multicast delay. This, however, is achieved at the expense of unicast cells. In particular, when the inputs have a load greater than 0.8, unicast cells experience high delays. We can reduce their delay by adjusting the parameter S_m, whereby we dedicate more time slots to unicast cells. Unfortunately, doing so increases the delay of multicast cells. The proposed scheduler, however, does not have the above limitations. The delay experienced by unicast cells is comparable to iSLIP-Emulate. On the other hand, even though multicast cells using our proposed scheduler have a slightly worst delay than SCIA, our scheduler does not cause severe performance degradation to unicast cells.

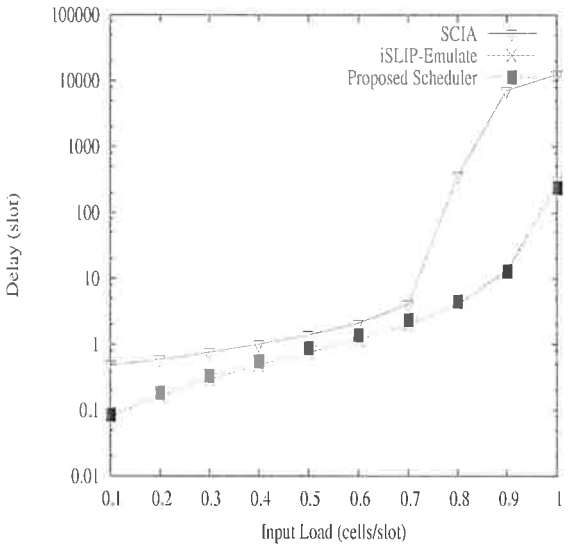

Figure 8: Average delay of unicast cells. Results are for an 3x3 switch with uniform i.i.d Bernoulli arrivals.

5.2 Non-Uniform Traffic

Using the same 8x8 switch, we change inputs' arrival to non-uniform Bernoulli traffic. Each input has a random load to each output that ranges from 0.0 to 0.1. As before, we designate half of the traffic to be multicast.

From Figures 10 and 11, we see that the proposed scheduler yields the best delay for both unicast and multicast cells. Comparatively, SCIA and iSLIP-Emulate have higher delays because both of them are known to have poor performance when traffic is non-uniform [16][8]. Intuitively, if a subset of inputs have a high unicast and multicast load, these schedulers will not consider these cells in the same round. For example, in SCIA, in a multicast time slot, it will try to maximize the number of multicast matchings without any regards to inputs with higher weighted unicast cells. In contrast, our scheduler considers both cell types in the same round, and schedules only the highest weighted cells. Moreover, it does not try to maximize the number of matchings unless multiple outputs deem a multicast cell to have the highest weight amongst all HOL cells that are destined for them.

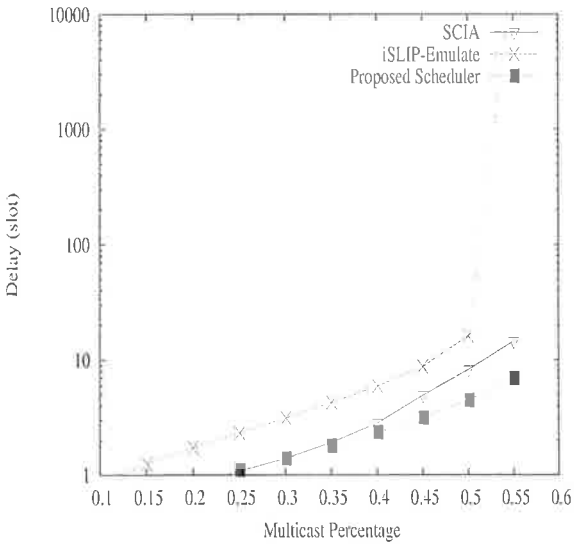

Figure 10: Average delay of unicast cells with non-uniform i.i.d Bernoulli arrivals.

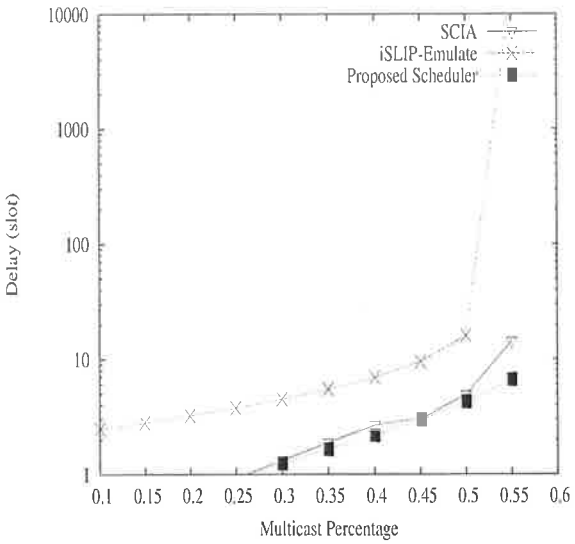

Figure 11: Average delay of multicast cells with non-uniform i.i.d Bernoulli arrivals..

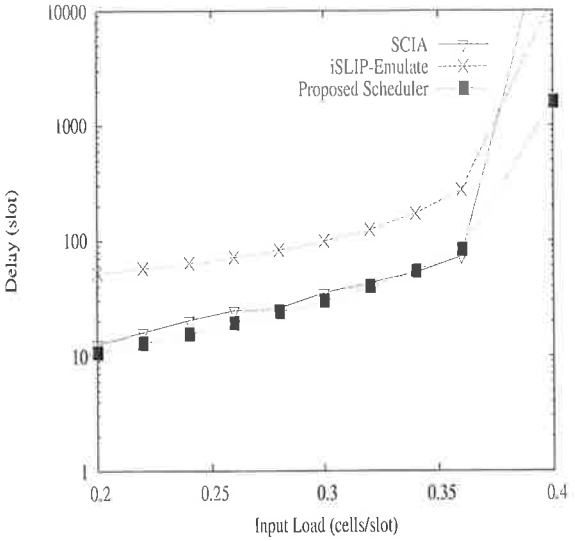

Figure 13: Average delay of multicast cells. Results are for an 8x8 switch with an arrival burst length of 10 cells.

5.3 Uniform Bursty Traffic

We now experiment with bursty traffic on a 8x8 switch. We start with uniform bursty traffic, where we increase the load of each input from 0.20 to 0.40 at an increment of 0.02. Moreover, we set the average burst size to 10 cells, and designate half the traffic to be multicast. Note, larger burst sizes simply cause a proportional increase in cell delay.

Figures 12 and 13 indicate that our scheduler has the lowest delay. SCIA has comparable delays, both for unicast and multicast cells, until the input load increases beyond 0.35. After such point, SCIA consistently prefers multicast over unicast cells, resulting in unicast queues becoming unstable. In comparison, the queues in our scheduler remain stable for a much higher input load, and treats both unicast and multicast cells fairly, as both cell types experience similar delays.

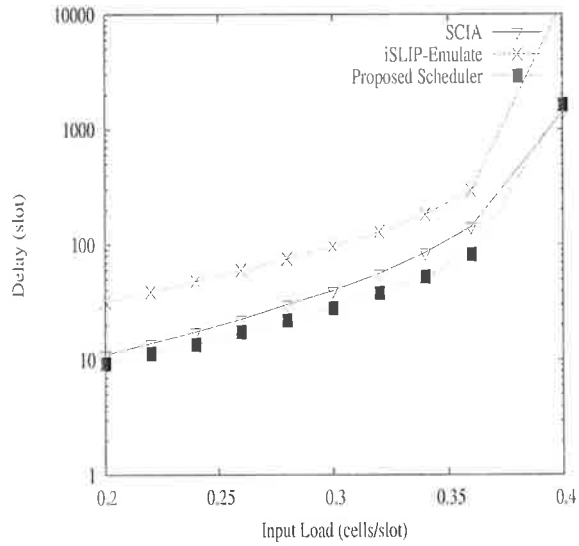

Figure 12: Average delay of unicast cells. Results are for an 8x8 switch with arrival burst length of 10 cells.

5.4 Non-Uniform Bursty Traffic

We continue the previous experiment but with non-uniform bursty traffic. Figures 14 and 15 show the same trend as the previous experiment. As the percentage of multicast cells increases, SCIA spends more time scheduling multicast cells at the expense of unicast cells. We could easily adjust S_m to reduce the delays of unicast cells by providing more opportunities to schedule them first. In practice, however, determining the best tradeoff is difficult as different flows at different points in time will have varying delay requirements.

5.5 Throughput

A key performance metric is a scheduler's throughput. From Figure 16, we see that our scheduler achieves 100% throughput when traffic is uniform and non-uniform; a significant advantage over iSLIP-Emulate and SCIA, given that they achieve 100% throughput only when traffic is uniform.

5.6 Switch Size

Lastly, we investigate how a switch's size, i.e., the number of inputs and outputs, impact our scheduler's performance. Figure 17 shows the delays incurred by unicast cells on a 8x8, 16x16 and 32x32 switch. We omit the plot for multicast cells because the delays are similar given that our scheduler treats both traffic types fairly. We see that our scheduler's performance degrades with increasing switch sizes. In a 32x32 switch, when the input load reaches 0.12, there is a significant increase in delay. This is due to multicast cell's large fanout. In fact, a multicast cell can have up to 32 outputs! We have also experimented with smaller fanout sizes. Our results indicate delays of cells for all switch sizes increase proportionally to the load and number of outputs.

6 Discussion

Our integrated scheduler performs better than existing approaches because of the following reasons:

Firstly, it considers the weight of both unicast and multicast cells simultaneously. This is in contrast to existing schemes that have thus far considered both traffic types separately. From our experimentations, we found this to

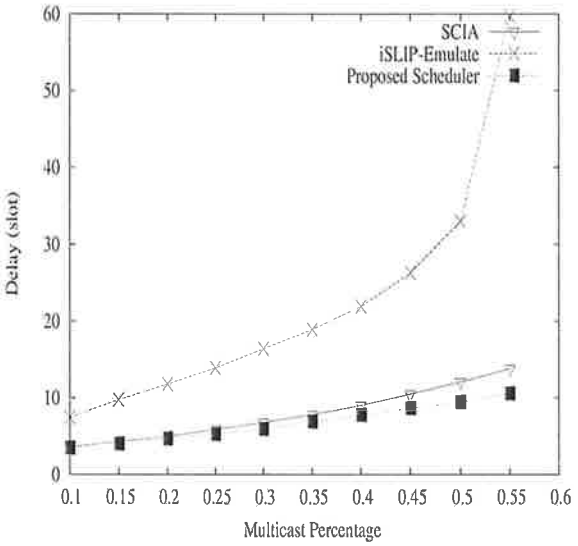

Figure 14: Average delay of unicast cells with non-uniform bursty traffic.

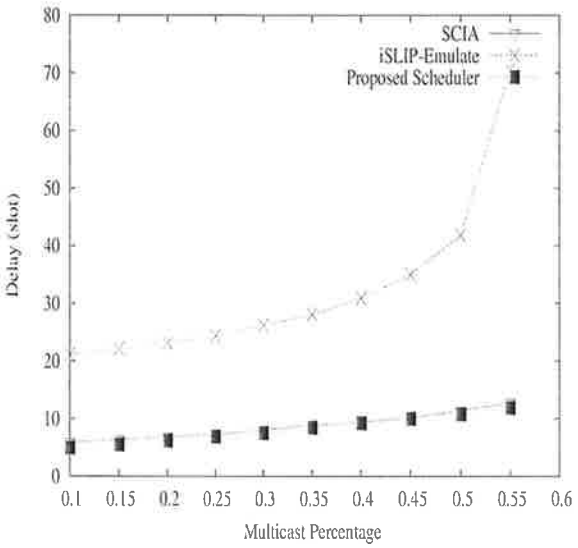

Figure 15: Average delay of multicast cells with non-uniform bursty traffic.

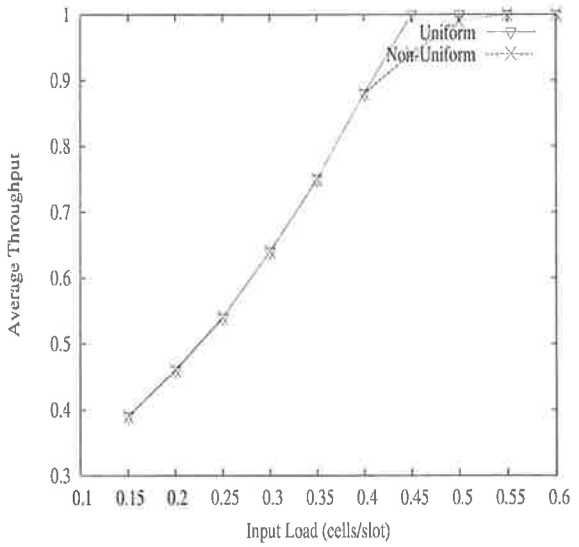

Figure 16: Average Throughput. Results are for an 8x8 switch with uniform or non-uniform traffic.

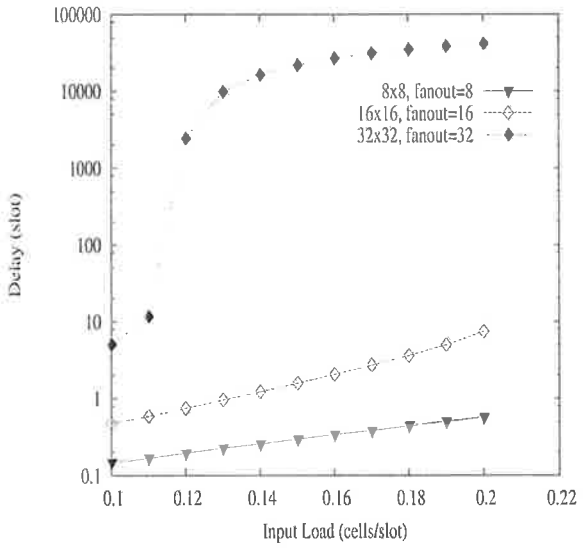

Figure 17: Average delay of unicast cells in different switch sizes; all inputs have uniform i.i.d Bernoulli traffic.

be critical when both unicast and multicast cells are competing for the same output. In SCIA, a tradeoff will have to be made as to which cell type should receive service. Moreover, this decision is not deterministic, as S_m designates a slot to be unicast/multicast probabilistically. Our scheduler, however, bases its decisions on cells' weight. Thereby, as our results showed, both unicast and multicast cells have the same delay.

Secondly, it uses staging buffers to address the multicast cell HOL blocking problem. Unlike existing works that utilize $k < 2^N - 1$ queues, our approach utilizes much less memory. The tradeoff, however, is extra computations involving TCAM lookups. Fortunately, these computations can be pipelined and is not critical to the scheduling process.

Thirdly, it utilizes the crossbar fabric's innate multicast ability opportunistically. Prior works such as [16] and [2] establish multicast matchings without considering the weight of unicast cells. Our scheduler, however, looks at both cell types and only enables the crossbar's multicast capability when multiple outputs deem a multicast cell to be the highest weighted cell in a given round. This is particularly advantageous as it increases a switch's throughput.

Fourthly, it does not use a tuning knob, e.g., S_m. From our results, we see that when inputs have low loads, our scheduler has comparable performance to SCIA [16]. However, we need to take into consideration that SCIA's performance is achieved by adjusting S_m iteratively. Our scheduler, however, does not have this limitation. Hence, it is able to operate with changing traffic conditions.

Lastly, it supports both uniform and non-uniform traffic. Existing approaches, such as iSLIP-Emulate [11] and SCIA [16], are designed for uniform traffic. Our scheduler, however, works well in non-uniform traffic scenarios. Specifically, in each round, it considers cells' weight, thereby allowing it to adapt to input loads that vary over time.

7 Conclusions

We have presented a novel scheduler capable of scheduling both unicast and multicast cells simultaneously. From our extensive simulation studies, our scheduler demonstrates comparable or better performance than existing schemes during low loads, and superior performance during high loads. More importantly, our scheduler is adaptive to changing traffic conditions, thereby making it suitable for both uniform and non-uniform traffic conditions.

References

[1] Internet 2 multicast applications. http://multicast.internet2.edu/wg-multicast-applications.shtml.

[2] M. Andrews, S. Khanna, and K. Kumaran. Integrated scheduling of unicast and multicast traffic in an input-queued switch. In *IEEE Infocom*, New York, USA, June 1999.

[3] A. Bianco, P. Giaccone, E. Leonardi, F. Neri, and C. Piglione. On the number of input queues to efficiently support multicst traffic in input queued switches. In *IEEE Workshop on High Performance Switching and Routing*, Torino, Italy, June 2003.

[4] M. Carol, M. Hluchyj, and S. Morgan. Input versus output queueing on a space division switch. *IEEE Transactions on Communications*, 35:1347–1356, Jan. 1988.

[5] J. Chao and B. Liu. *High Performance Switches and Routers*. Wiley-Interscience, 2007.

[6] M. A. Marsan, A. Bianco, P. Giaccone, E. Leonardi, and F. Neri. Multicast traffic in input-queued switches: Optimal scheduling and maximum throughput. *IEEE Transactions on Networking*, 11(3):465–477, June 2003.

[7] N. McKeown. SIM: A fixed length packet simulator. http://klamath.stanford.edu/tools/SIM/.

[8] N. McKeown. *Scheduling Algorithms for Input-Queued Cell Switches*. PhD thesis, Department of Electrical Engineering, University of California Berkeley, may 1995.

[9] N. McKeown. A fast switched backplane for a gigabit switched router. *Business Communications Review*, 27(12):1020–1030, 1997.

[10] N. McKeown. The iSLIP scheduling algorithm for input-queued switches. *IEEE Trans. on Networking*, 7(2):188–198, Apr. 1999.

[11] C. Minkenberg. Integrating unicast and multicast traffic scheduling in a combined input and output queued packet switching system. In *IEEE ICCCN*, pages 127–134, Las Vegas, USA, Oct. 2000.

[12] B. Prabhakar, N. McKeown, and R. Ahuja. Multicast scheduling for input-queued switches. *IEEE Journal on Selected Areas in Communications*, 15(5):855–866, June 1997.

[13] E. Schiattarella and C. Minkenberg. Fair integrated scheduling of unicast and multicast traffic in an input-queued switch. In *IEEE ICC*, Istanbul, Turkey, June 2006.

[14] K. Schultz and P. Gulak. CAM-based single-chip shared buffer ATM switch. In *IEEE Conference on Communications*, New Orelans, USA, May 1994.

[15] M. Song and W. Zhu. Throughput analysis for multicast switches with multiple input queues. *IEEE Communications Letters*, 8(7):479–481, 2004.

[16] W. Zhu and M. Song. Integration of unicast and multicast scheduling in input-queued packet switches. *Elsevier Computer Networks*, 50(8):667–687, Aug. 2006.

Conferences in Research and Practice in Information Technology

Volume 107

Parallel and Distributed Computing 2010

Australian Computer Science Communications, Volume 32, Number 6

AUSTRALIAN COMPUTER SOCIETY

Parallel and Distributed Computing 2010

Proceedings of the Eighth Australasian Symposium on Parallel and Distributed Computing (AusPDC 2010), Brisbane, Australia, January 2010

Jinjun Chen and Rajiv Ranjan, Eds.

Volume 107 in the Conferences in Research and Practice in Information Technology Series.
Published by the Australian Computer Society Inc.

Published in association with the ACM Digital Library.

Parallel and Distributed Computing 2010. Proceedings of the Eighth Australasian Symposium on Parallel and Distributed Computing (AusPDC 2010), Brisbane, Australia, January 2010

Conferences in Research and Practice in Information Technology, Volume 107.

Editors:

Jinjun Chen
Faculty of Information and Communication Technologies
Swinburne University of Technology
1, Alfred Street, Hawthorn,
Melbourne, Victoria 3122
Australia
Email: jchen@swin.edu.au
Rajiv Ranjan
School of Computer Science and Engineering
The University of New South Wales
Sydney, NSW 2052
Australia
Email: rranjans@gmail.com

Series Editors:
Vladimir Estivill-Castro, Griffith University, Queensland
Simeon J. Simoff, University of Western Sydney, NSW

crpit@scm.uws.edu.au

Publisher: Australian Computer Society Inc.
PO Box Q534, QVB Post Office
Sydney 1230
New South Wales
Australia.

Conferences in Research and Practice in Information Technology, Volume 107.
ISSN 1445-1336.
ISBN 978-1-920682-88-0.

Printed, December 2009 by UWS Press, Locked Bag 1797, South Penrith DC, NSW 1797, Australia
Document engineering by Susan Henley, University of Western Sydney
Cover Design by Matthew Brecknell, Queensland University of Technology
CD Production by FATS Digital, 318 Montague Road, West End QLD 4101, http://www.fats.com.au/

The *Conferences in Research and Practice in Information Technology* series aims to disseminate the results of peer-reviewed research in all areas of Information Technology. Further details can be found at http://crpit.com/.

Table of Contents

Contributed Papers

Preface

These proceedings contain the papers presented at the 8th Australasian Symposium on Parallel and Distributed Computing (AusPDC 2010), held on 18 January in Brisbane Australia in conjunction with the Australasian Computer Science Week (ASCW). Over the years, previously known as Australasian Symposium on Grid Computing and e-Research (AusGrid), and starting this year, it is being referred to as AusPDC, has become the flagship symposium for Grid, Cloud, Cluster, and Distributed Computing research in Australia. In total, 16 submissions were received, mostly from Australia, but also from New Zealand, United States, Asia and Europe. The full version of each paper was carefully reviewed by at least three referees, and evaluated according to its originality, correctness, readability and relevance. A total of 8 papers were accepted. The accepted papers cover topics from cloud resource management, grid inter-operation, multi-processing systems, trusted brokering, performance models, operating systems, and networking protocols.

We are very thankful to the Program Committee members, and external reviewers for their outstanding and timely work, which was invaluable for taking the quality of this year's program to such a high level. We also wish to acknowledge the efforts of the authors of all paper submissions, without whom this conference would not be possible. Due to the very competitive selection process, several strong papers could not be included in the program. We sincerely hope that prospective authors will continue to view the AusPDC symposium series as the premiere venue in the field for disseminating their work and results. We would also like to thank the ACSW organizing committee, those that submitted papers and those that attended the conference their work and contributions have made the symposium a great success.

Rajiv Ranjan
University of New South Wales

Jinjun Chen
Swinburne University of Technology

AusPDC 2010 Programme Chairs
January 2010

Programme Committee

Chairs

Jinjun Chen, Swinburne University of Technology, Australia
Rajiv Ranjan, University of New South Wales, Australia

Members

Jemal Abawajy, Deakin of University, Australia
David Abramson, Monash University, Australia
Mark Baker, University of Reading, UK
David Bannon, Victoria Partnership for Advanced Computing, Australia
Boualem Bentallah, University of New South Wales, Australia
Rajkumar Buyya, University of Melbourne, Australia
Paul Coddington, University of Adelaide, Australia
Neil Gemmell, University of Otago, New Zealand
Andrzej Goscinski, Deakin University, Australia
Kenneth Hawick, Massey University, New Zealand
John Hine, Victoria University of Wellington, New Zealand
Jane Hunter, University of Queensland, Australia
Martin Johnson, Massey University, New Zealand
Nick Jones, University of Auckland, New Zealand
Laurent Lefevre, University of Lyon, France
Andrew Lewis, Griffith University, Australia
Anna Liu, University of New South Wales, Australia
Piyush Maheshwari, Perot Systems, USA
Teo Yong Meng, National University of Singapore, Singapore
Manish Parashar, Rutgers University, USA
Srikumar Venugopal, University of Melbourne, Australia
Yun Yang, Swinburne University of Technology, Australia

Steering Committee

Prof. David Abramson, Monash University, Australia
Prof. Rajkumar Buyya, University of Melbourne, Australia
Dr. Jinjun Chen (Vice Chair), Swinburne University of Technology, Australia
Dr. Paul Coddington, University of Adelaide, Australia
Prof. Andrzej Goscinski (Chair), Deakin University, Australia
Prof. Kenneth Hawick, Massey University, New Zealand
Prof. John Hine, Victoria University of Wellington, New Zealand
Dr. Rajiv Ranjan, University of NSW, Australia
Dr. Wyne Kelly, Queensland University of Technology, Australia
Prof. Paul Roe, Queensland University of Technology, Australia
Dr. Andrew Wendelborn, University of Adelaide, Australia

Organising Committee

Co-Chairs

Dr. Wayne Kelly
Prof. Mark Looi

Budget and Facilities

Mr. Malcolm Corney

Catering and Booklet

Dr. Diane Corney

Sponsorship and Web

Dr. Tony Sahama

Senior Advisors

Prof. Colin Fidge
Prof. Kerry Raymond

Finance and Travel

Ms. Therese Currell
Ms. Carol Richter

Registration

Mr. Matt Williams

DVD and Signage

Mr. Matthew Brecknell

Satchels and T-shirts

Ms. Donna Teague

Welcome from the Organising Committee

On behalf of the Australasian Computer Science Week 2010 (ACSW2010) Organising Committee, we welcome you to this year's event hosted by the Queensland University of Technology (QUT). Striving to be a "University for the Real World" our research and teaching has an applied emphasis. QUT is one of the largest producers of IT graduates in Australia with strong linkages with industry. Our courses and research span an extremely wide range of information technology, everything from traditional computer science, software engineering and information systems, to games and interactive entertainment.

We welcome delegates from over 21 countries, including Australia, New Zealand, USA, Finland, Italy, Japan, China, Brazil, Canada, Germany, Pakistan, Sweden, Austria, Bangladesh, Ireland, Norway, South Africa, Taiwan and Thailand. We trust you will enjoy both the experience of the ACSW 2010 event and also get to explore some of our beautiful city of Brisbane. At Brisbane's heart, beautifully restored sandstone buildings provide a delightful backdrop to the city's glass towers. The inner city clusters around the loops of the Brisbane River, connected to leafy, open-skied suburban communities by riverside bikeways. QUT's Garden's Point campus, the venue for ACSW 2010, is on the fringe of the city's botanical gardens and connected by the Goodwill Bridge to the Southbank tourist precinct.

ACSW2009 consists of the following conferences:

- Australasian Computer Science Conference (ACSC) (Chaired by Bernard Mans and Mark Reynolds)
- Australasian Computing Education Conference (ACE) (Chaired by Tony Clear and John Hamer)
- Australasian Database Conference (ADC) (ADC) (Chaired by Heng Tao Shen and Athman Bouguettaya)
- Australasian Information Security Conference (AISC) (Chaired by Colin Boyd and Willy Susilo)
- Australasian User Interface Conference (AUIC) (Chaired by Christof Lutteroth and Paul Calder)
- Australasian Symposium on Parallel and Distributed Computing (AusPDC) (Chaired by Jinjun Chen and Rajiv Ranjan)
- Australasian Workshop on Health Informatics and Knowledge Management (HIKM) (Chaired by Anthony Maeder and David Hansen)
- Computing: The Australasian Theory Symposium (CATS) (Chaired by Taso Viglas and Alex Potanin)
- Asia-Pacific Conference on Conceptual Modelling (APCCM) (Chaired by Sebastian Link and Aditya Ghose)
- Australasian Computing Doctoral Consortium (ACDC) (Chaired by David Pearce and Rachel Cardell-Oliver).

The nature of ACSW requires the co-operation of numerous people. We would like to thank all those who have worked to ensure the success of ACSW2010 including the Organising Committee, the Conference Chairs and Programme Committees, our sponsors, the keynote speakers and the delegates. Special thanks to Justin Zobel from CORE and Alex Potanin (co-chair of ACSW2009) for his extensive advice and assistance. If ACSW2010 is run even half as well as ACSW2009 in Wellington then we will have done well.

Dr Wayne Kelly and Professor Mark Looi
Queensland University of Technology

ACSW2010 Co-Chairs
January, 2010

CORE - Computing Research & Education

CORE welcomes all delegates to ACSW2010 in Brisbane. CORE, the peak body representing academic computer science in Australia and New Zealand, is responsible for the annual ACSW series of meetings, which are a unique opportunity for our community to network and to discuss research and topics of mutual interest. The original component conferences ACSC, ADC, and CATS, which formed the basis of ACSWin the mid 1990s now share the week with seven other events, which build on the diversity of the Australasian computing community.

In 2010, we have again chosen to feature a small number of plenary speakers from across the discipline: Andy Cockburn, Alon Halevy, and Stephen Kisely. I thank them for their contributions to ACSW2010. I also thank the keynote speakers invited to some of the individual conferences. The efforts of the conference chairs and their program committees have led to strong programs in all the conferences again, thanks. And thanks are particularly due to Wayne Kelly and his colleagues for organising what promises to be a strong event.

In Australia, 2009 saw, for the first time in some years, an increase in the number of students choosing to study IT, and a welcome if small number of new academic appointments. Also welcome is the news that university and research funding is set to rise from 2011-12. However, it continues to be the case that per-place funding for computer science students has fallen relative to that of other physical and mathematical sciences, and, while bodies such as the Australian Council of Deans of ICT seek ways to increase student interest in the area, more is needed to ensure the growth of our discipline.

During 2009, CORE continued to work on journal and conference rankings. A key aim is now to maintain the rankings, which are widely used overseas as well as in Australia. Management of the rankings is a challenging process that needs to balance competing special interests as well as addressing the interests of the community as a whole. ACSW2010 includes a forum on rankings to discuss this process. Also in 2009 CORE proposed a standard for the undergraduate Computer Science curriculum, with the intention that it be used for accreditation of degrees in computer science.

COREs existence is due to the support of the member departments in Australia and New Zealand, and I thank them for their ongoing contributions, in commitment and in financial support. Finally, I am grateful to all those who gave their time to CORE in 2009; in particular, I thank Gill Dobbie, Jenny Edwards, Alan Fekete, Tom Gedeon, Leon Sterling, and the members of the executive and of the curriculum and ranking committees.

Justin Zobel

President, CORE
January, 2010