Time-Slotted Scheduling for Agile All-Photonics Networks: Performance and Complexity

Hana Bilbeisi

Department of Electrical & Computer Engineering
McGill University
Montreal, Canada

September 2007

A thesis submitted to McGill University in partial fulfillment of the requirements for the degree of Masters of Engineering.

2007/09/29

Your file Votre référence
ISBN: 978-0-494-51448-1
Our file Notre référence
ISBN: 978-0-494-51448-1

Abstract

Schedulers in optical switches are still electronic, the performance of these units has a significant impact on the performance of the network and could form a bottleneck in high speed networks, such as AAPN. Four time-slotted scheduling algorithms are investigated in this study, PIM, iSlip, PHM and Adapted-SRA. The study addresses the performance of AAPN for each of the algorithms, and evaluates the hardware complexity, estimating the running time of the algorithms. Performance measures were collected from an OPNET model, designed to emulate AAPN. Furthermore, hardware complexity and timing constraints were evaluated through hardware simulations, for iSlip, and through analysis for the rest of the algorithms. iSlip confirmed it's feasibility by meeting the $10us$ timing constraint set by AAPN. The study revealed the superiority of iSlip and PHM over PIM and Adapted-SRA.

Sommaire

Les 'planifiant' dans les commutateurs optiques est toujours lectronique, l'excution de ces units a un impact significatif sur la ralisation du rseau et pourrait former un goulot d'tranglement dans les rseaux rapides, tels que AAPN. Quatre types d'algorithmes d'ordonnancement a rpartition dans le temps sont examins dans cette tude, PIM, iSlip, PHM et l'Adapt-SRA. L'tude adresse l'excution de AAPN, l'valuation de la complexit de 'hardware' et l'estimation du temps courant pour chacun des ces algorithmes. Les mesures d'excution ont t recueillies d'un modle de OPNET, conu pour imiter AAPN. De plus, la complexit de 'Hardware' et la synchronisation du system ont t values par des simulations, pour iSlip, et par l'analyse pour le reste des algorithmes. La mthode iSlip a affirm sa praticabilit en ralisant la 10S contrainte mis par l'AAPN. L'tude a rvl la supriorit de iSlip et PHM par comparaison au PIM et l'Adapt-SRA.

Acknowledgments

The work presented in this thesis would not have been done without the aid of many people. My supervisor, Professor Lorne Mason, and the staff in the Engineering Department, provided me with the support, knowledge and guidance that enabled me to complete this work.

I am particularly grateful for my family for instilling in me confidence and supporting my educational pursuits, especially my twin sister who has been always there for me. I am also thankful for all my friends for their encouragement.

Finally, I would like to acknowledge the organizations that funded the AAPN project, giving me and many other researchers the opportunity to participate in such an emerging research area.

Contents

List of Figures

List of Tables

List of Acronyms

2DRR	2Dimentional Round-Robin
AAPN	Agile All-Photonic Network
ASIC	Application Specific Integrated Circuit
Bps	Bits per Second
BS	Burst Switching
DRRM	Desynchronized Round-Robin Matching
FIFO	First In First Out Queues
FIRM	First-Come-First-Serve in Robin Matching
FPGA	Field Programmable Gate Arrays
FSM	Finite State Machine
HOL	Head Of Line
IQ	Input Queuing
LAN	Local Area Network
LRD	Long Range Dependence
MAN	Metropolitan Area Network
OBS	Optical Burst Switching
OQ	Output Queuing
OTDM	Optical Time Division Multiplexing
OWDM	Optical Wave Division Multiplexing
PHM	Parallel Hierarchical Matching
PIM	Parallel/Probabilistic Iterative Matching
PRRM	Probabilistic Round-Robin Matching
RDSRR	Rotating Double Static Round-Robin
RRM	Round-Robin Matching

SRA	Single Round-Robin Arbitration
SRD	Short Range Dependent
TDM	Time Division Multiplexing
VoD	Video On Demand
VoIP	Voice Over IP
WAN	Wide Area Network
WDM	Wave Division Multiplexing
WWFA	Wrapped wave form arbiter

Chapter 1

Introduction

1.1 Introduction Photonic networks:

The telecommunication world is witnessing an enormous increase on the demand for bandwidth due to the emergence and rapid growth of broadband services. The evolution of network services is facing challenges ranging from performance reliability, scalability and resource utilization, to service provisioning and profitability. The introduction of optical networks resolved performance and utilization challenges. Network agility on the other hand, explores dynamic service provisioning in attempt to maximize profitability.

Transmission through optical fibers brings about several advantages, like large bandwidth, immunity to noise and interference, and low costs per unit bandwidth. Thereby, optical networks provide high capacity, supporting high bandwidth services. The definition of optical networks refers to the transmission through optical-fibers; the terms *optic* and *photonic* are used interchangeably in this thesis. Existing networks involve the integration of optical networks and electrical switching, which requires conversion between the optical and electrical domains upon switching. The conversion process forms a bottleneck in such hybrid networks. Furthermore, developments in optical technologies introduced optical switches that are transparent to data format and bit rate, and have greater switching capacity than electronic switches. All-optical networks utilize optical switches, where both transmission and switching take place in the optical domain, thereby avoiding the conversion bottleneck. However, several tasks concerning buffering, addressing and labeling are not supported in the optical domain, which complicates the design and implementation of

the network.

The popularity of broadband applications, for example VoD and VoIP, is creating unpredictable network environments. Dynamic network configuration and automated service provisioning are required to support such environments through a cost-efficient approach. The term Agile, in Agile All-Photonic Networks (AAPN), refers to the ability to optimize network operation in a dynamic fashion,*dynamic reconfigurability*.

1.2 AAPN

AAPN is a research program that targets the exploitation of agility in all-photonic networks, as the name implies [1]. The program is structured into three themes: Networks and Architecture, Enabling Technologies and System Integration. The research done in this thesis contributes to the first theme, where it investigates the application of scheduling techniques that best suit the network requirements. The second theme follows the developments of optical technologies, and finally the third combines the findings in the first two themes.

Traffic enters and leaves AAPN through edge nodes; ingress edges represent source nodes, while egress edges represent destination nodes. Edge nodes are connected through a crossbar core switch. The switch connects ingress and egress nodes to enable transmission between networks connected to AAPN.

The extension of data paths of all-photonic networks, to reach within close distances of end-users, comprises the motivation of AAPN's first theme [1]. Network agility refers to the ability of the network to perform multiplexing for dynamic allocation of the bandwidth to traffic flows. In AAPN, agility is achieved by employing resource sharing methods; OBS or OTDM, to WDM.

1.3 TDMA and time-slotted scheduling

Xiao et al.[2] Investigated the performance of AAPN under two modes of resource sharing; OBS and two classes of OTDM. Slot-by-slot scheduling and frame based OTDM

comprise the two classes of OTDM. The study revealed the superiority of OTDM over OBS in terms of performance measures. Slot-by-slot achieved better performance in MANs, while frame-based OTDM performs well in both MANs and WANs, but requires complex signaling. Slot-by-slot scheduling is adopted in the study present in this thesis.

1.4 Objective of the study

A scheduler resides in the control unit of the core switch, where it runs a matching algorithm to configure the crossbar inter-connects. The scheduling process steers the performance of the network, and could form a bottleneck in such high speed networks. Performance attributes of the network, and algorithms' running times, form the basic criteria for evaluating schedulers in AAPN. This thesis presents a study of several time-slotted algorithms, addressing the criteria mentioned above. Research was first conducted to nominate a number of schedulers for the study. The application of each of the nominated schedulers was further simulated using an OPNET[1] model that emulates AAPN [3], to evaluate the effect on the network's performance. Finally the speed of convergence of one of the algorithms was simulated through a hardware model design. Timing assessment of the rest of the algorithms was done through associating results from the simulation model and those reported in the literature review.

Maximum matching algorithms find the maximum number of matches possible in a certain event. On the other hand, maximal matching algorithms [4] explore an iterative approach, where the number of possible matches increases with the number of iteration runs. Maximum matching proved to be optimal in terms of performance but complex in comparison to maximal matching algorithms.

Nonetheless, some maximum matching algorithms, like SRA proved to be less complex than others, while achieving the same level of performance [5]. PIM was one of the first maximal matching algorithms to be considered. The application of PIM [4] to AAPN was demonstrated in previous work [3] [2]. Xiao [3] also proposed a modified version of PIM and evaluated its effect on AAPN. RRPM is the basic Round Robin arbitration algorithm. The algorithm iSlip [6], a modified version of RRPM, has grown to be a research standard in the scheduling literature. Later on, diverse variants of arbitration schedulers started

[1] A networking simulation tool

emerging like DRRM [7], RDSRR [8], FIRM [9], offering some performance improvements. Hierarchical matching forms a third category of maximal matching algorithms, PHM [10] [11] belongs to this category. PIM, iSlip, PHM and SRA were adopted for performance evaluation in the AAPN environment, using the OPNET model.

Switch controllers utilize ASICs or FPGA chips to run their scheduling algorithms. The study of the hardware requirements and actual running time of a scheduler can be achieved by implementing its functionality on FPGA chips, due to their applicability and low cost. Reference [12] briefly discussed the basic hardware design of iSlip's protocol in embedded systems. Moreover, studies conducted by McKeown and Gupta [13] involving the Tiny Tera project[2] [14] , tackled several aspect of the hardware design of iSlip like algorithms. McKeown and Gupta [13] presented the optimal solution proving that the algorithm meets the requirements of Tiny Tera. Finally, the running time estimation of PHM was discussed in [15] without any design specifications.

The simulation model used in this study employs a Cyclone II FPGA chip. It adopts a simplified design that improves on the model in [12] [16] and exploits the results in [13]. The results obtained from running the simulation model for different numbers of nodes are used as an analysis reference to assess the running time of the rest of the algorithms.

1.5 Outline of the thesis

The following chapter provides the reader with the necessary background regarding the architecture of AAPN, the slot-by-slot scheduling process and the characteristics of the researched scheduling algorithms. Chapter 3 describes the tools utilized in the study, presenting a complete model derivation for both the performance and hardware complexity simulations. Results obtained from both models are illustrated in chapter 4. The chapter provides a comprehensive discussion of the results. The thesis concludes with a summary of the basic results and proposals for future work in chapter 5.

[2]Tiny Tera is a packet switch with a switching capacity close to 1 Tera bps

Chapter 2

Background

Network services are developing to keep pace with the booming popularity of broadband applications. The development of services is facing challenges ranging from performance reliability, scalability and resource utilization, to resource allocation and profitability. Furthermore, the evolution of network technologies is progressively facilitating the resolution of such challenges.

The maturation of optical technologies supports optical networking, which consequently opens new doors for networking advancements and architectures. Photonic networks provide ideal performance, utilization and scalability solutions. Network agility explores capabilities offered by engineering approaches, to employ dynamic resource allocation in an attempt to maximize profitability. An introduction of an agile all-photonic network is outlined in this chapter.

One should note that networking technologies are simply tools, and that their employment practices have a great influence on their effectiveness. Optical switches comprise a good example in the context of our study, where their performance is affected by the network architecture, topology, and scheduling schemes.

The chapter starts by presenting the AAPN research project, which structures the framework of this thesis. The second section briefly discusses time-slotted transmission and resource sharing techniques, presenting conclusions drawn from former studies regard-

ing scheduling in AAPN. The last section provides a detailed description of scheduling algorithms, and their adaptation to conform to the AAPN environment.

2.1 Agile All Photonics Network

AAPN is a research network funded by the Government of Canada's Natural Sciences and Engineering research Council (NSERC), and other Canadian companies and laboratories [1]. The project was launched in 2003 on a five year agenda.

Communication networks were initially purely electronic, and then evolved to form a hybrid of electronics and photonics. Hybrid networks involve optical transmission and electronic switching, where domain conversion is required from optical to electrical upon switching, and back to optical before transmitting. Finally the increasing demand for bandwidth induced the introduction of all-photonic networks, currently a major concern in the field of telecommunications. Transmission and switching in all-photonic networks take place in the optical domain, eliminating the need for any kind of conversion between domains. All-optical networks utilize optical switches that are capable of handling data at higher speeds than electronic switches, which indicates that switching[1] is faster and the conversion bottleneck (OEO) is avoided. Optical switches are transparent to data format and bit rate, which facilitates processing and configuration. On the other hand, several tasks like buffering, addressing and labeling are not supported in the optical domain, which complicates the design and implementation of the network.

AAPN's main motivation is to extend the data path of all-photonic networks as close as possible to the end-users side [1]. Such an objective could be achieved by network agility. Network agility refers to the ability of the network to perform multiplexing for dynamic allocation of the bandwidth to traffic flows. In AAPN, agility is achieved by employing resource sharing techniques OTDM or OBS to WDM.

[1]Assuming the technology of all-optical-space switches will reach a point to support high capacities, high port-count and fast-reconfiguration[17].

2.1.1 Network Design

The topology of the network has a significant effect on its implementation and performance. Several aspects were taken into consideration in designing the layout of AAPN. Major concerns are explained below:

1. Application:
   Being a core network, AAPN requires a robust topology, one that would accommodate for the occurrence of faults and continue to distribute traffic loads over a large number of switches. Mesh topologies are most suitable for core networks.

2. Capacity:
   Photonic core switches have huge capacity which could be suppressed by the employment of a mesh topology. Efficient capacity exploitation would be achieved through scalable and less complex topologies, typically tree or star.

3. Control of All-photonic switches:
   Due to the lack of buffering and other packet switching tasks, meticulous control functionality is required to resolve potential contentions among ingress nodes. The coordination of the switches would be very complex in a mesh topology. Again star topologies are far more appropriate for reducing the cost and complexity of the control challenge.

Vickers and Bashai [18] proved that overlaid star topologies outperform mesh architectures in all-photonic networks, in a study launched for an earlier all-photonic network project, Petaweb. Mason et al.[19] addressed the problem of topological design in AAPN. The study investigated an overlaid star network topology, regarding several aspects: cost, capacity and traffic demand. Among the conclusions of the study was a confirmation of the aptness of the composite star topology in AAPN.

An overlaid star topology is depicted in figure 2.1, where a star layer forms the basic unit of the structure. The edge nodes in the overlaid stars are logically connected. The Logical mesh-like connection targets robustness by compensating for the point of weakness in star topologies.

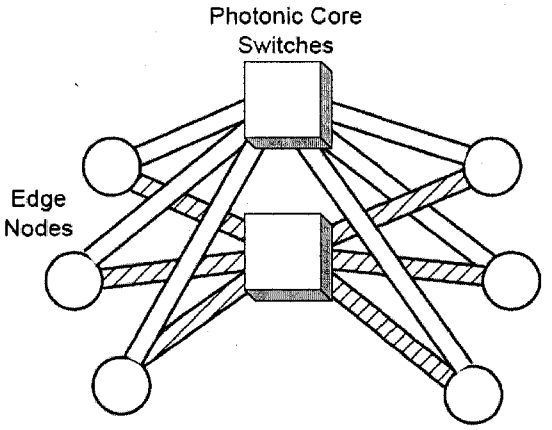


Fig. 2.1 Overlaid Star architecture:[17]

2.1.2 Network Architecture

AAPN consists of a multilayer star network refer to figure 2.1, a single star is illustrated in figure 3.3. One should note that wavelength conversion is not supported by AAPN. Edge nodes form the ingress/egress points to/from the network. Data transmission between two nodes takes place through a single star within a single wavelength. Thus a wavelength on an outgoing link of a photonic switch is allocated to only one connection through the network. As a result, there is no interaction between data amongst the stars, confirming the independence of the stars and the distributed control of the core switches.

In summary, data paths between edge nodes and the core switches are purely photonic. Conversion takes place at the edge nodes, from the electrical to optical domain and vice versa. Buffering takes place at the edge nodes and will be discussed in the following section.

2.1.3 Multi-Queue Buffers

Electrical cross bar switches employ input buffering, output buffering or a combination of both with a speed up factor. M. Karol et al.[20] provided a comparison between Input Queuing (IQ) and Output Queuing (OQ) in a packet switch. However, optical cross bar switches do not support queuing on either side of the switch. Therefore, queuing in AAPN takes place at the edge nodes.

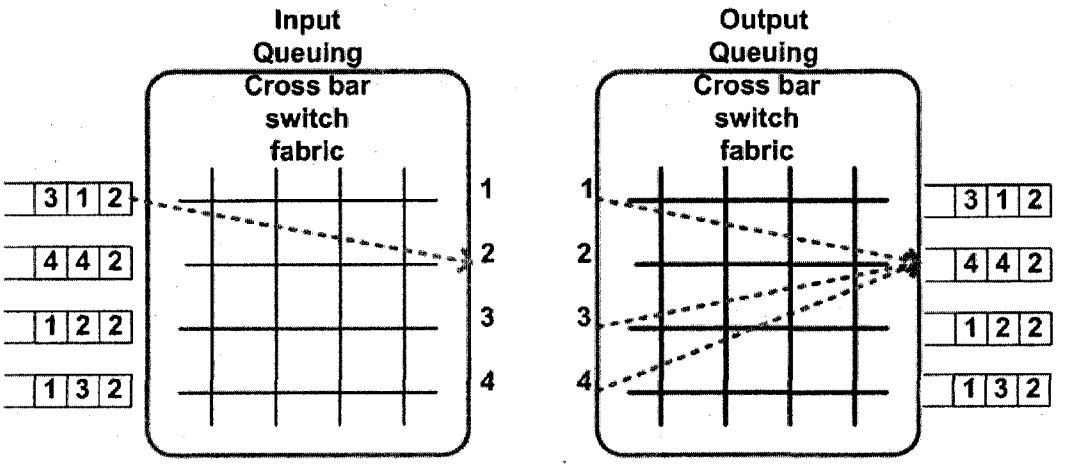


Fig. 2.2 IQ versus OQ in cross-bar switches

OQ exhibits several advantages over IQ [20], but involves simultaneous transmission of more than one packet through the switch. Figure 2.2 illustrates the requirement of transmitting up to N packets to the same output port in OQ, while only one packet could be sent in the IQ cross bar. Such a simultaneous transmission requires internal speed-up S in the cross bar fabric, S=4 in the figure. The speed up requirement complicates the implementation and memory requirements of the switch. As a result, cross bar switches usually employ IQ.

In a best effort environment, each input port utilizes a FIFO queue to buffer packets that are destined to any of the output ports. Head Of Line (HOL) blocking is a consequence of FIFO queuing. HOL blocking occurs when a packet destined to a certain free output port X is kept waiting because a packet ahead in the queue, destined to another output Y, is blocked. HOL blockage contributes to high latency delays and throughput deterioration. M. Karol et al. [20] proved that the HOL blockage limits the maximum throughput of a single IQ switch to 58%. Several techniques were suggested to mitigate the effect of HOL. Virtual output queuing, introduced by Tamir et al.[21] proved to be the most efficient approach. Virtual Output Queues (VOQs) form logical separations within a single buffer, each separation buffers packets targeting a single output port. By that, N VOQs are required in each input port, for an N x N network.

The term VOQ switch refers to the structure employing VOQ buffering in a cross bar switch. VOQ switches became really popular due to their high throughput and cheap implementation. The architecture of AAPN exploits the benefits of VOQs by employing them in each edge node. Figure 2.3 demonstrates the VOQ cross bar switch model employed in the study.

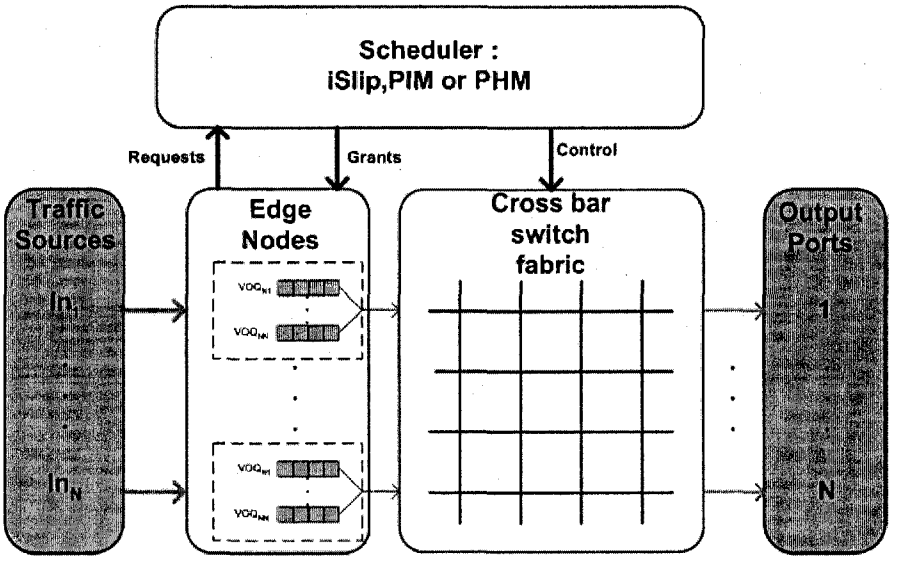


Fig. 2.3 VOQ cross-bar switch in AAPN

2.2 Agility through Resource Sharing Mechanisms

Agility in AAPN is enabled through the network topology and deployment of photonic switches. Photonic switches operate in the order of sub-microseconds, providing a huge margin for granularity in resource sharing. Moreover, the overlaid star topology supports the introduction of various resource sharing techniques, OBS and OTDM are examples of such techniques.

In OBS, traffic is assembled into bursts according to an aggregation technique, before being sent to the core switch. Whereas in OTDM, sources are allocated time slots through which they can send a specific amount of traffic. Therefore, the difference between the two schemes lies in the assembly and amount of traffic transmitted between the edge nodes and

the cross bar switch. The functionality of OTDM entails synchronization among the edges.

Overlaid star topologies support network synchronization, enabling the application of OTDM in AAPN. Moreover, simulations in [2] and others in [19], revealed that OTDM techniques are more robust to traffic variations in the network, which induced the adoption of OTDM techniques in AAPN.

Statistical slot-by-slot scheduling and frame based deterministic scheduling comprise two classes of OTDM. In slot-by-slot scheduling, request signals from the ingress nodes are used to reserve the output ports of the switch on a slot-by-slot basis. The speed by which slots are reserved is limited by the delay of signaling required to grant a reservation request, which in turn depends on the network coverage. Such a limitation deteriorates the performance making the application impractical in WAN topologies. Frame based scheduling is preferable in cases of large distance network coverage. In frame based scheduling, multiple slots are reserved according to traffic prediction techniques. The scheme is more complex than slot-by-slot scheduling, and so slot-by-slot scheduling is employed in this study.

2.3 Slot-by-Slot Scheduling Schemes

Crossbars are configured by electronic controllers that run scheduling algorithms. A controller runs the algorithm at the beginning of each time slot to resolve contention between service requests. The scheduling algorithm examines the set of service requests submitted by the N^2 VOQs, where N is the number of nodes in the network, and then forms a matching map between input and output ports. The concept is best described by the bipartite matching,as defined below. The algorithm is said to converge when the maximum number of outputs is matched to service requests.

The application of four scheduling algorithms to the AAPN design was studied and is presented in this thesis. The algorithms are PIM, as a continuation of a former study [3], iSlip, PHM and finally adapted-SRA. The following subsections provide an overview of each one of these algorithms.

2.3.1 Bipartite matching

The process of configuring a cross bar switch is equivalent to a matching problem. Matching problems involve undirected bipartite graphs. The mathematical definition of these graphs is outlined below, as illustrated by [22].

The undirected graph is: G (V, E)

- V: A finite set of nodes or vertices
- E: A finite set of edges
- Endpoints of an edge: Nodes that are attached to an edge M: is a match that acquaints a pair of nodes and an edge, where edges do not share common nodes
- Matched node: An endpoint of an edge in the matching

In a bipartite graph, the set of nodes can be divided into two disjoint and independent sets:
V_1 and V_2, where G:=(V_1 + V_2, E)

Condition: None of the edges could have both endpoints in the same set

A matching behavior could achieve one of the following:

1. Maximum Match:
   This is a matching approach that joins the maximum number of nodes, and consequently contains the largest possible number of edges in a specific event.

2. Maximal Match:
   Matching occurs in stages, where edges are added at every stage, if a match is present. An input in a maximal match could have one of two states: it could either be a part of a match, or all the outputs it requested are already matched. It should also be noted that every maximum match is maximal.

3. Complete Match:
   Matching that covers all the nodes in the graph. A complete match is maximal and maximum.

Application to the Core Switch

The input nodes represent one set of the nodes (V_1) The output nodes represent the other set (V_2) The crossbar interconnections represent the edges.
Figure 2.4 views a bipartite graph demonstration of the crossbar switch, along side its matrix presentation. The matrix derived from the graph presents the format of the request sent to the scheduler.

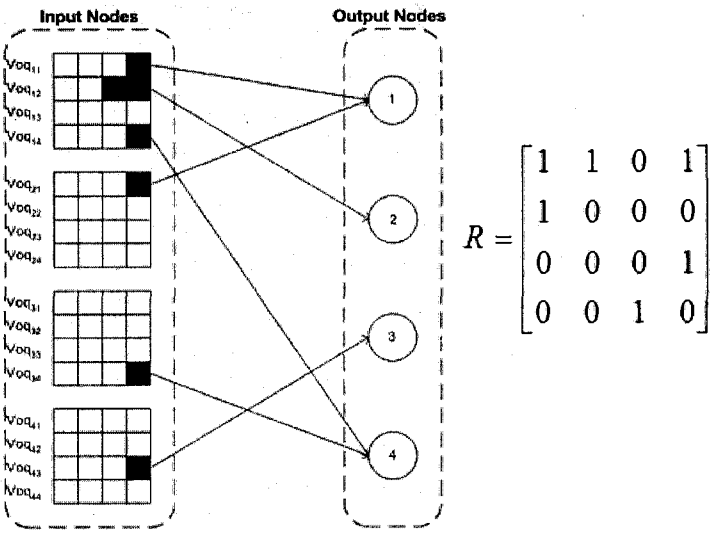


Fig. 2.4 Bipartite Graph

2.3.2 Scheduling schemes characteristics

Different scheduling algorithms acquire different characteristics, ranking their suitability for the requirements of a certain design. The following is a set of characteristics that should be taken in consideration when choosing a scheduling scheme:

1. Performance measures:
   (a) Utilization and Throughput
   (b) Delay
   (c) Loss rate

2. Scalability:
   The implementation complexity in cross-bar switches is of the order $O(N^2)$, N being the number of ports. Such a characteristic would be further emphasized by employing a scheduler that does not scale well. Scalability is a major concern in the choice of a scheduling scheme.

3. Starvation of nodes:
   A node is said to starve for service when none of its submitted requests get served, causing the input queues to overflow and subsequently increasing the rate of packet loss. Starvation of nodes in a crossbar is mainly due to the scheduling behaviour

4. Fairness of the matching:
   The flow of some algorithms tend to discriminate between input/output ports, on the service level. Such algorithms impose unfairness in the network.

5. Computational complexity:
   The computational complexity of a scheduler influences its hardware requirements and the speed by which the algorithm runs. While speed is one of the most important factors in the AAPN design, the scheduler could form a bottleneck in the core. The design assigns up to 10us to the scheduling operation. A scheduler that requires more than the allocated interval would not fit the design requirements.

6. Hardware Requirements:
   Schedulers are implemented in hardware, usually ASICs or FPGAs, the following factors should be considered in hardware design:

   (a) Simplicity of implementation: Complex schedulers require off-chip communication, making it more expensive and complex.

   (b) Speed requirement

   (c) The area occupied on chip

   (d) Memory requirements

   (e) Power consumption

   (f) Pipelining amendments for better processing utilization

2.4 Matching behaviors with respect to the requirement characteristics of a scheduling scheme

2.4.1 Maximum Match

Achieving a maximum match leads to the highest link utilization and throughput among other matches. However, the complexity of finding a maximum match for an N x N crossbar is O(N(N+M)), M being the number of edges [4]. Such a high performance complexity results in speed deterioration, causing high service delay and starvation of the ports. It also indicates that the scheduling algorithm is not scalable.

More efficient maximum-size bipartite matching algorithms have been proposed in the literature. Single Round-Robin Arbitration (SRA)[5] is investigated in this study as a contribution of maximum-size matching algorithms. The original SRA algorithm does not suit the AAPN design, for reasons explained in the next section, so an adapted version is proposed.

SRA

As the name implies, SRA employs a single round-robin arbiter for each of the crossbar output ports. Arbiters are used to select inputs that are matched in a time slot. There are different arbitration schemes, these are further explored in the maximal matching algorithms' section, and are explained in detail below.

The original SRA algorithm as described in [5], is not iterative and finds up to N matches in a single time slot. The algorithm uses a dynamic FIFO queue for each output arbiter. Each queue keeps status records of the inputs' corresponding VOQ, so the queues could be up to size N. For example, if VOQ_{ij} has queued cells, output j will have an entry for input i in its status queue. An output node chooses the value at the head of its queue. The arbiter then grants service to that input and removes it from the head of the queue. If the VOQ still has queued packets, it sends a request and gets added to the tail of the queue, otherwise it loses its spot in the queue.

The algorithm could match more than one VOQ within an input, allowing the input to send to more than one output in a single time slot. However, a single port in an optical switch can not be involved in more than one matching in a time-slot, on a single wavelength. Thus the original SRA algorithm should be modified to fit the AAPN design.

Adapted SRA is a modified version of SRA and is employed in the AAPN simulation model for the performance study.

Adapted SRA has the same functionality as SRA, except that matching in adapted SRA is done between single input/output ports. When an output arbiter sends a grant to an input port, it waits for an acceptance/rejection message, if the input has been matched to another output, it sends a rejection message. If the output receives an acceptance message it follows the original SRA protocol. On the other hand, if it receives a rejection message, it adds the port's element to the tail of the queue and grants service to the port that appears next in the queue, and the process repeats.

Properties of the Adapted SRA:

1. Complexity:
   SRA and adapted SRA are not iterative, but adapted SRA has a complexity of $O(N^2)$. The Simulations showed that the algorithm is very slow in comparison to other matching algorithms explored in this study.

2. Scalability:
   This property tackles two aspects, the hardware requirements of the algorithm, and the amount of signaling or communication messaging involved in the protocol.

   (a) Hardware requirements: only one set of arbiters is involved in the implementation of SRA, whereas iSlip and PIM require two sets. On the other hand, Adapted SRA involves N queues of a maximum size N, requiring a controller with larger memory.

   (b) Communication messages: The original SRA requires fewer messages than those used in iterative maximal matching algorithms, like iSlip and PIM, discussed below. Adapted SRA involves extra messaging between the nodes, but still requires less messaging than maximal matching algorithms.

3. Fairness:
   Adapted SRA is a fair algorithm based on the utilization of the FIFO status queues. The technique of adding the status element of a matched port to the end of the FIFO queue indicates that the port gets the least priority in the next time slot.

2.4.2 Maximal Match

Link utilization of maximal matches is much worse than that of the maximum matches; in fact it could get to low of 50% depending on the matching algorithm.

The complexity of the algorithms vary, but proved to be much less than that exhibited by maximum matches. Thus maximal matching algorithms are more flexible.

A variety of maximal matching algorithms were introduced in the literature. Maximal matching is iterative, where the scheduling optimization problem converges to a local maximum after running a certain number of iterations. Probabilistic matching algorithms, like the Parallel Iterative Matching algorithm, PIM [4], is one of the first maximal matching algorithms. RRPM is the basic Round Robin arbitration algorithm that descended from PIM. iSlip[6] which is a modified version of RRPM, became a research standard in the scheduling literature. Thereafter, different flavors of arbitration schedulers emerged,such as DRRM[7], RDSRR[8], FIRM[9], offering performance improvements and supporting QoS. A third scheduling category involves hierarchical matching techniques [10] [23] [24].

Xiao et al. proposed a modified version of PIM in [2] [3]. The modifications were based on improving the overall performance of the network. Whereas Pan and Yang.[25] and McKeown and Gupta [13], proposed amendments on existing maximal matching algorithms targeting hardware efficiency and timing constrains.

Parallel Iterative matching

Parallel iterative matching is a maximal matching algorithm. It randomly chooses the endpoints of each edge in the bipartite graph. The algorithm employs independent arbiters that select nodes in a probabilistic fashion.

Simulations done in [2], [3], [4], and this thesis illustrate that this algorithm yields link utilization in the range of 85% to 100%, depending on the probability function utilized by the arbiters, and the number of iterations run by the algorithm.

It has been claimed that randomness reduces the number of iterations required to achieve the maximal match [26]. That however is dependent on the random generation process. For example, if all grants were given to the same input, only one match would be performed in that iteration, and N iterations would be required to reach a maximal match. On the other hand, if every granted input is unique, the algorithm would converge in one iteration. However, on average PIM matches 3/4 of the potential matches in each iteration and thus

the algorithm converges to a maximal match in $O(\log_2 N)$ iterations, the mathematical proof is provided in [4]. The algorithm is starvation free, which is also due to and dependent on the random generation process.

Random matching algorithms have some drawbacks. The first is that of the hardware complexity of the algorithm. Each arbiter is supposed to run a random number generation function, which is highly expensive in terms of hardware [26]. According to scheduling hardware measures, a tradeoff is usually made between storage elements and processing time. Unlike other algorithms, PIM does not require memory elements to hold the state of the matches done in the past. However random generation requires a considerable amount of time, in hardware. Processing time is more important in the AAPN design, making PIM disadvantageous in that field. Furthermore, adding extra nodes to the network means adding extra random generators that are already expensive, which indicates that the algorithm is not scalable

Fairness is another concern in PIM [6] [4] [26]. Since the selection process done by each node is completely random and independent, nodes will have different admission probabilities, leading to unfairness among nodes, especially when the nodes are oversubscribed. Figure 2.5 illustrates an example of unfairness in PIM [26]. The figure demonstrates the discrimination between ports. Several solutions were introduced by [3] [4]. Traffic monitoring in [3] resolves the matter by limiting the submission of a service request to a certain number of packets. While that does not resolve the matter completely, it does mitigate unfairness. Weighted matching would be a more effective solution, which requires more processing and storage of past connection states, violating the basic properties of PIM. Thus applying weighted matching to PIM adds to the hardware complexity while improving its overall performance.

One last problem with PIM is that it does not perform well when running only one iteration. For a single iteration, the performance of PIM is comparable to FIFO switches, where it achieves a throughput around 63% (refer to the results chapter). As mentioned before PIM converges when run for $(\log_2 N)$ iterations, but that requires a high rate of operation.

The algorithm takes place through three stages: Request, Grant and Accept:

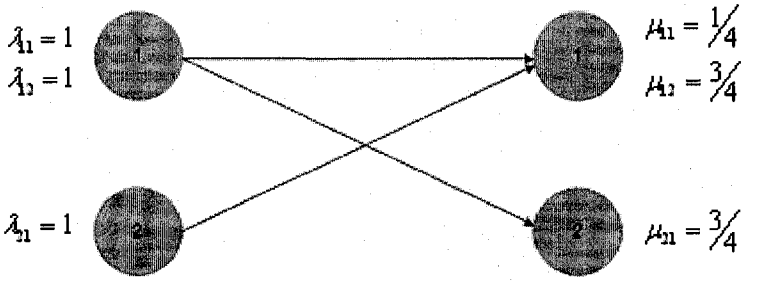


Fig. 2.5 Discrimination among ports in PIM

Request: Each unmatched input submits a service request for each output for which it holds queued cells.

Grant: Each unmatched output selects one input request, randomly among all the requests it receives (if any), and grants service to it.

Accept: Each input selects one output grant randomly, if it receives any.

Previous Work :

Xiao studied the performance of PIM in the AAPN environment, and proposed a modified version of the algorithm, called the Adapted-PIM, that performs better under the design constraints [3]. Adapted-PIM tackled performance and fairness aspects.

The modification was based on improving performance measures in the network. Adapted-PIM involves a set of memory elements that store requests which are not serviced in an iteration run; these are called left-requests. For example, if two service requests out of four are serviced in the first iteration, the other two requests gets stored in memory for the second iteration. Requests in the left-request registers get submitted in the next iteration. An input port keeps on submitting its request until it is serviced, meaning that in PIM, service requests experience the round-trip propagation delay over and over until they are granted service. Whereas the introduction of left-request queues saves the need of generating and sending new requests, mitigating the propagation delay.

Adapted-PIM introduced another modification on PIM's performance measures, mainly link utilization, called fill-up matching. Fill-up matching accounts for the possibility of missing a potential match after running several iteration runs of the algorithm. Basically the algorithm passes over each and every one of the un-matched nodes checking for a matching possibility. That proved to greatly enhance link utilization and throughput, but increases the processing time of the algorithm.

Adapted-PIM tackled the issue of fairness by setting a boundary limit on the number of packets buffered in the VOQ before a request is sent to the core. The request-boundary mitigates the chance of randomly matching an input with only one packet, while other inputs' VOQs are full and about to overflow.

Adapted PIM proved to improve performance measures in the network. However, the modifications incur extra hardware and require longer processing time, which does not conform to the constraints of AAPN. Therefore, the original version of PIM is used in this study with the addition of employing left-request registers.

iSlip

iSlip is an iterative matching algorithm derived from the functionality of PIM. Both algorithms follow the same three staged protocol and arbitration concepts. However, the actual scheduling of input/output differs. iSlip employs rotating priority (Round Robin) arbitration instead of randomness in matching ports. Furthermore, the algorithm enhances the basic operation of RRM to achieve better performance. The arbitration process will be illustrated before discussing the properties of the algorithm. Since iSlip is a variation of the RRM algorithm, the scheduling operation of RRM is outlined first.

RRM devotes an arbiter for every input/output port. The algorithm follows the protocol outlined in the operation of PIM, but instead of employing randomness in selecting the ports, it utilizes a round robin scheduler. A single iteration run of the round robin scheduler follows the steps outlined below:

Request: Each unmatched input submits a service request for each output for which it holds queued cells.

Grant: If an unmatched output receives requests, it selects the first input that appears after the one pointed at by g_i and sends it a grant. g_i is then incremented (modulo N) to one location beyond that of the granted input.

Accept: if an unmatched input receives any grants, it selects the first one that appears after the output pointed at by a_j. The pointer is then incremented (modulo N) to one location beyond that of the accepted output.

Key:
g_i: The grant pointer of the arbiter of output[i]
a_j: The grant pointer of the arbiter of input [j]
i,j: {0,...,N}

The variation between RRM and iSlip lies in the fashion by which g_i is updated. In iSlip g_i is only updated in the first iteration, if and only if requesting input accepts the grant from output[i]. a_j are updated the same way in both algorithms [6]. The Grant stage is the only source of difference between the algorithms. Figure 2.6 presents an example that demonstrates the operation of matching through arbitration in iSlip. The figure presents the operation of the algorithm's stages for two iterations.

Grant: If an unmatched output receives any requests, it selects the first input that appears after the one pointed at by g_i and sends it a grant. If the algorithm is in the first iteration, and an accept is received by the granted input, g_i is incremented (modulo N) to one location beyond that of the granted input. Otherwise the grant pointer would not move.

The fashion by which pointers are updated in iSlip has the effect of desynchronizing the grant arbiters under certain traffic loads. Moreover, the desynchronization of arbiters increases the rate by which the algorithm converges, the phenomena was further explored by DRR[8]. Figure 2.7 demonstrates the effect of high traffic loads on the synchronization of grant arbiters.

The behavior of iSlip brings about the following properties[6]:

1. The algorithm is starvation free:
   An input i would keep on requesting service until it is granted. Furthermore, an output j would serve up to (N-1) inputs before reaching i, where it might need to wait up to

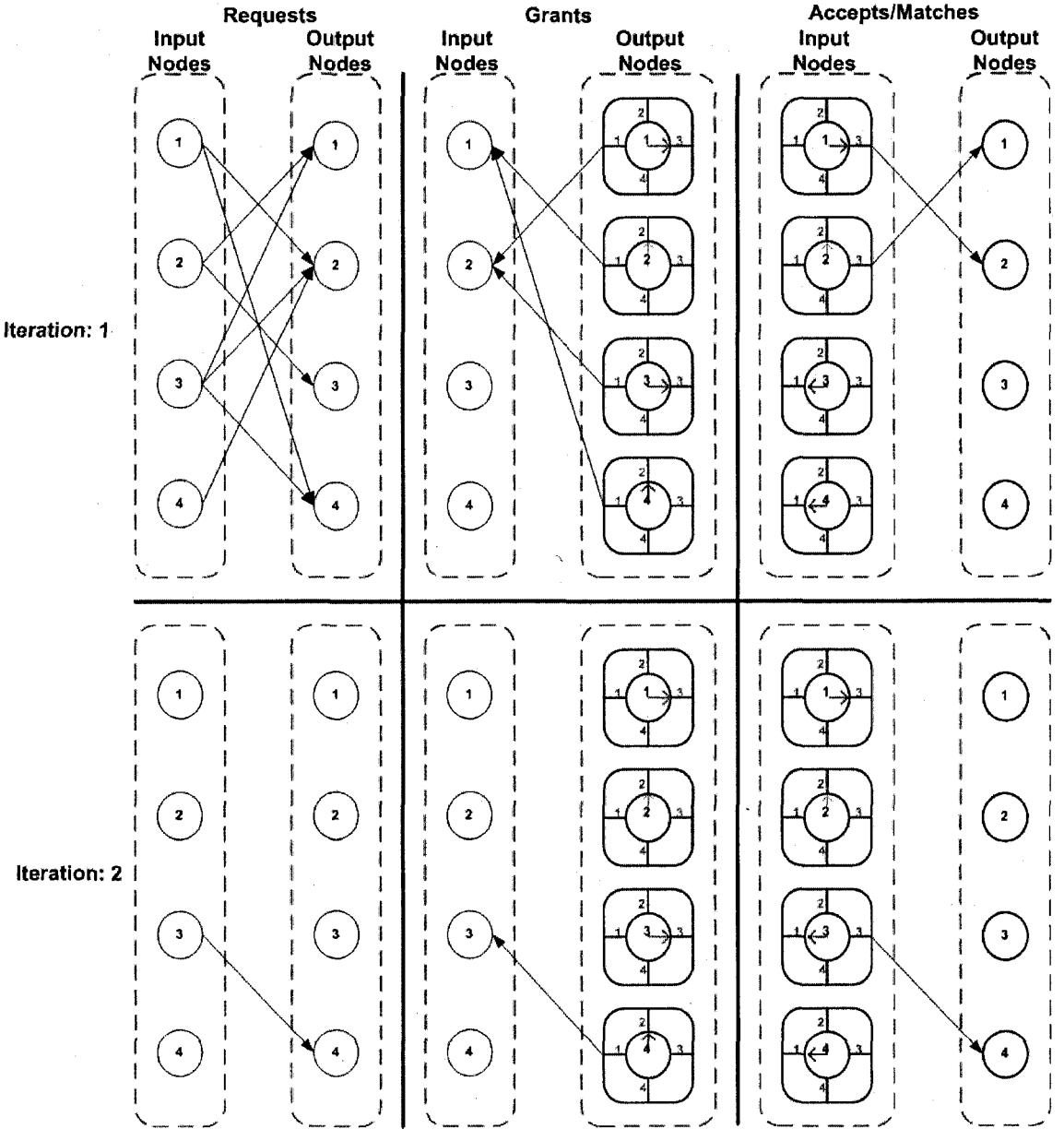


Fig. 2.6 Arbitration and matching in iSlip

N time-slots to be accepted (since the granting pointer would not select another input otherwise). By that input i will certainly be served in a time frame less than N^2 slots.

2. The algorithm is fair:
   Connections made in the first iteration have the lowest priority in consecutive time-slot, which is a consequence of the arbitration regime. The algorithm does not discriminate between input/output ports, since the selection takes place in a fixed order.
3. Speed of convergence:
   The speed of convergence in iSlip depends on several factors, mainly the offered load. At high offered load the algorithm produces a large amount of matches and might even converge in a single iteration, O(1). However, analytical studies showed that iSlip converges in at most N iterations under regular traffic load.

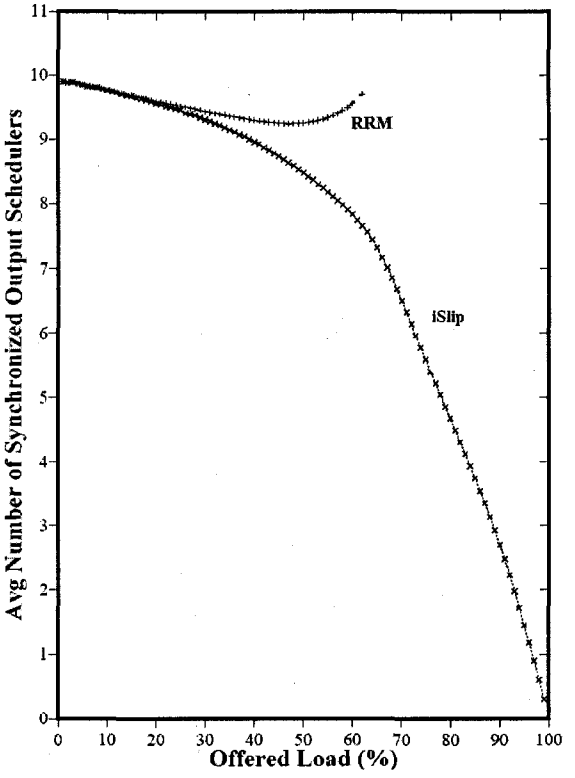


Fig. 2.7 The Effect of traffic load on the synchronization of arbiters[6]

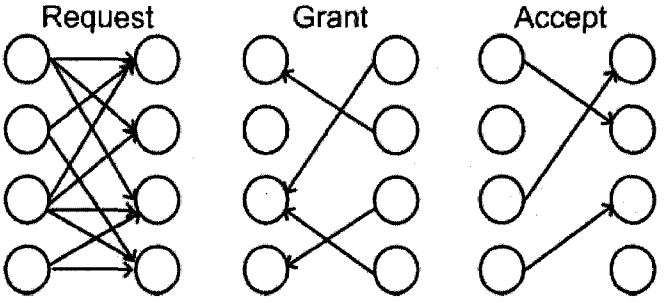
Notes about the hardware requirements of PIM and iSlip :

PIM and iSlip follow the same matching protocol, three-step iterative matching. The protocol requires the exchange of about $\{(N^2 + 2\,N)\log_2 N\}$ messages. N^2 request messages from each VOQ, N grant messages and N accept messages. Furthermore the total number of messages is multiplied by the number of iterations by which the algorithm is supposed to converge, the equation assumes that it is $\log_2 N$ on average[2]. Each of the messages contain $\log_2 N$ bits, which accounts for the hardware requirements of the scheduler, mainly the number of the I/O pins, memory, on-chip area, and power consumption. So far the discussion has disregarded the hardware requirements of the implementation of arbitration; that is discussed in detail in chapter 4.

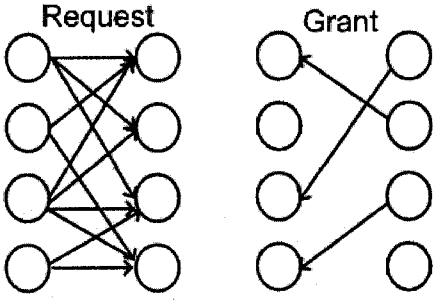
Peng and Yang [25] proposed a hardware efficient two step iterative matching algorithm for VOQ switches. The proposed algorithm incorporates arbitration into the request step and eliminates the accept step. The arbitration in the request step selects one request service from each of input ports, and sends it to the grant step. However, it should be noted that sending a single request indicates that the number of grants to be received gets limited to a maximum of one, eliminating the need for the accept step. Figure 2.8 demonstrates the idea of two step matching algorithms. The figure shows a matching problem resolved through three step matching (a), and through two step matching (b). The algorithm has a shorter scheduling time and requires fewer message exchanges. Analytical studies and simulations presented in [25] show that the rate of convergence of the proposed algorithm is close to that of the three-step algorithm.

McKeown and Gupta [13] proposed a pipelined implementation of the three-step iterative matching protocol in iSlip. The implementation pipelines the grant step of iteration i with the accept step of iteration i+1. A sequential flow of iterations seems to be the only possible approach, due to the dependence of the grant step of one iteration on the accept step of the previous iteration. The grant step uses the feedback from the accept step to identify the matched inputs, so that it would disregard their requests in the following iterations, as seen in 2.6. However, the functionality of the grant step of iteration i could be partially governed by the grant step of iteration i+1. To further elaborate, it is known that an input will definitely be matched upon receiving at least one grant, so if the grants produced in one iteration are ORed together and fed to the grant step of the second iteration,

[2]Simulations show that this figure is less in iSlip



(a) Three Step Matching



(b) Two Step Matching

Fig. 2.8 Operation Stages of Iterative PIM/iSlip

that grant step will disregard further requests from the granted inputs, without the need for any further input. The pipelining implementation decreases the number of clock cycles required to run a number of iterations of the algorithm. For the study outlined in [13] the number of clock cycles were reduced from 4i to 2i+2. The original implementation requires four clock cycles per iteration, two clock cycles for each step while joining the request and grant steps.

PHM

Hierarchical matching algorithms form a different class of schedulers, based on maximum size guesses [10]. The algorithms operate by dividing the VOQs into N maximum throughput groups. Each of the groups is assigned to a unit hierarchy in the system. Matching is done with respect to the hierarchical level of each VOQ, acting as a priority measure.

2DRR[24], WFA and WWFA[23] are algorithms in that class. The algorithms demonstrate variations of the basic arbitration matching process. Figure 2.9 below, illustrates the operation by which the basic WFA and its enhanced version (WWFA) blend arbitration and hierarchical techniques. In WFA, an arbitration wave propagates through a set of arbiters, the order by which these arbiters are traversed sets their level in the hierarchy. WWFA follows the same strategy as WFA, but employs a different arbitration wave, as depicted by the figure. WWFA converges faster by increasing the number of arbiters in each class. Moreover, 2DRR is a generalization of WFA and WWFA, the algorithms follow the same arbitration through classes mechanism. However, 2DRR enhances fairness by altering the pattern by which the arbiters are categorized, every time slot.

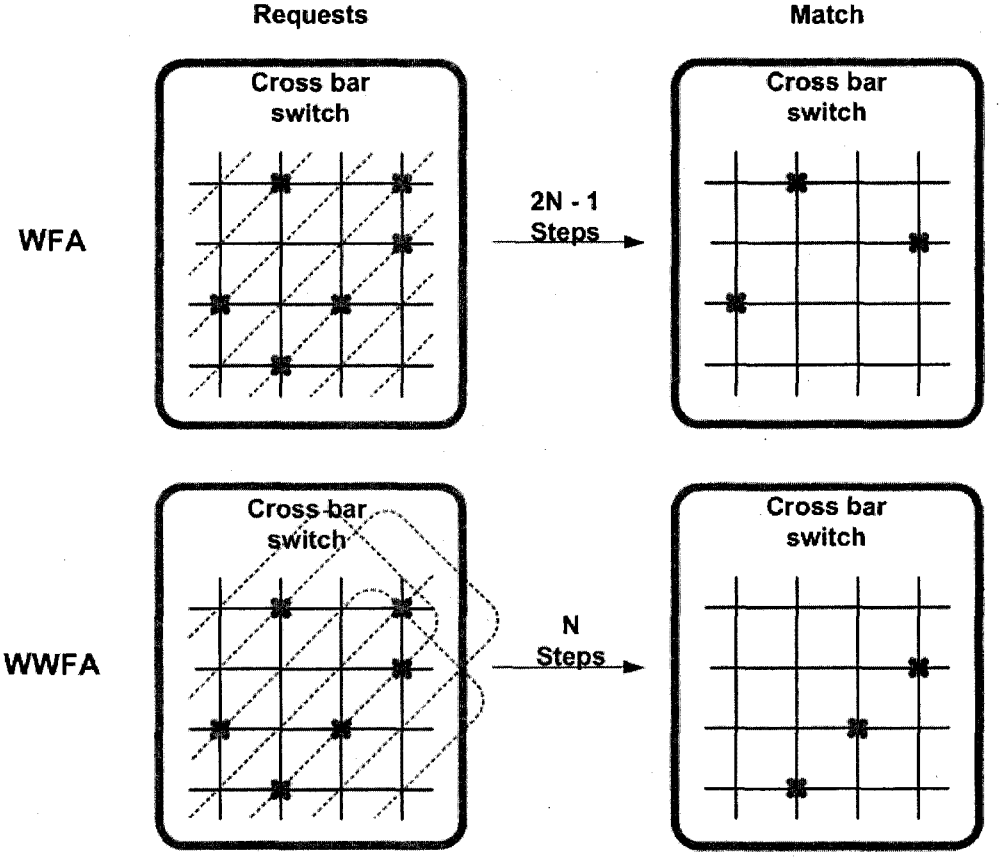


Fig. 2.9 WFA vs WWFA

2DRR and WWFA require a maximum of O(N) iterations to converge, and so are

called sequential hierarchical matching algorithms. A parallel hierarchical matching (PHM) scheduler was introduced by [10] [11].

A hierarchy matrix H in PHM is used to divide the VOQs into different levels. The matrix is then associated with all the service requests (arranged in a request matrix) to form the matching. Below is presentation of the PHM algorithm along side the definition of the variables used, as outlined in [11]. Finally an example demonstrating the matching process in PHM is provided.

Definitions in the PHM algorithm:

- r_{ij} i,j={1,.....,N} : Indicates the submission of a request from VOQ(i,j)
  $\Rightarrow$ *if* r_{ij}=1 then VOQ(i,j) is requesting service, otherwise VOQ(i,j)is empty s_{ij} i,j={1,.....,N} : indicates that VOQ(i,j) has has been selected for transmission at the current time-slot.
- h_{ij} : Hierarchical unit of s_{ij}
- t_{ij} : Auxiliary variable used to break the inter-dependence among the groups in sequential hierarchical matching.

Outline of the PHM algorithm:

1. Initialization of variables:
   s_{ij}^0=0 AND n=0, where n is the number of iterations
2. DO
   (a) Do in parallel:
   IF:
   $r_{ij} = 1$ *AND*
   $\forall k \neq i, s_{kj}^n$=0 AND
   $\forall k \neq j, s_{ik}^n$=0
   then t_{ij}=1 else t_{ij}=0
   (b) Do in parallel
   IF:
   $t_{ij} = 1$ *AND*
   $\forall k \neq i \mid h_{kj} > h_{ij}, t_{kj}$=0 AND
   $\forall k \neq j \mid h_{ik} > h_{ij}, t_{ij}$=0 AND
   $s_{ij}^{n+1} = 1$
3. n=n+1
   IF: n $\neq$ Number of iterations
   THEN: Go to Step 2
   ELSE: End

Tabe 2.1 presents a detailed workout through a PHM example, given a hierarchical matrix H and a request matrix R.

The behaviour of PHM depends on the routine by which the hierarchical matrix is updated. Several routines were suggested in the literature. Updating routines must take in consideration the nature of the traffic, and the application of the scheduler.

Xaio [3] discussed the topic of scheduling for differentiated services in AAPN. Furthermore, the discussion led to the proposal of additional features to the adapted PIM to support a class based implementation. Hierarchical matching algorithms support QoS

Table 2.1 PHM-Example

$H = \begin{bmatrix} 3 & 2 & 1 & 0 \\ 2 & 1 & 0 & 3 \\ 1 & 0 & 3 & 2 \\ 0 & 3 & 2 & 1 \end{bmatrix}$ $R = \begin{bmatrix} 1 & 0 & 1 & 1 \\ 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 \\ 0 & 1 & 0 & 0 \end{bmatrix}$ for two iterations step1: initialize S^0 to a zero matrix

Iteration 1, n=0	Iteration 2, n=1
$t^1 = \begin{bmatrix} 1 & 0 & 1 & 1 \\ 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 \\ 0 & 1 & 0 & 0 \end{bmatrix} \Rightarrow S^1 = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \end{bmatrix}$	$t^2 = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 1 & 0 & 0 \end{bmatrix} \Rightarrow S^2 = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 1 & 0 & 0 \end{bmatrix}$
Output: The Matching Matrix=S^2	

through class based implementations in a more natural way, avoiding the need for additional hardware and software complications in the system, while maintaining high levels of performance

F. J. Gonzlez-Castao et al.[11] provided an analytical comparison between the timing constraints set by PHM and other iSlip-like algorithms. The study in [11] also presented timing results obtained from implementing the two classes of algorithms using Ambit ASIC technology. Moreover, Soto et al.[15] evaluated the hardware requirements of PHM by implementing its functionality on a FPGA chip. The results obtained from both studies illustrated faster timing responses in PHM.

Chapter 3

Simulation Model

This chapter presents the study of the proposed schedulers in AAPN. The study branches to simulate the performance of the schedulers on one hand, and the hardware implementation, and timing measures on the other hand. An OPNET [27] model was utilized to test the performance of each of the schedulers in the AAPN environment. Furthermore, the hardware implementation of iSlip was evaluated by implementing its functionality on an FPGA chip. An association between the simulated results and the characteristics of iSlip is further exploited to evaluate the hardware and timing requirements of the rest of the schedulers.

The first part of the chapter depicts the AAPN performance model. This part starts by outlining performance measures collected from the OPNET model. It also discusses the exploitation of certain traffic patterns in the design. The simulation model is then described in detail. The second part demonstrates the hardware design of the iSlip scheduler.

3.1 Performance Simulation Model

The objective of this part is to evaluate the performance of the network as a result of employing different schedulers. The performance of the network is not the only concern, since the actual hardware implementation of some scheduling algorithms could be impractical. However, this part of the study disregards the practicality of the hardware implementation and focuses on the performance of the network.

In previous work [3], one layer of the AAPN star architecture was modeled in OPNET. The model collects performance measures resulting from the employment of different sched-

ulers under different traffic patterns. The scheduling algorithms discussed in chapter 2 were coded in the simulation model, for evaluation and comparison purposes.

3.1.1 Model Derivation

The design of a simulation model is based on three elements: the input injected into the model, the processing operation, and the output of the processing. Defining these elements is very important in the design process.

The processing operation is defined by the AAPN networking problem, the main concern at this stage is the scheduler. A set of proposed schedulers were discussed in the previous chapter. Moreover, the traffic injected into the network has a significant impact on its performance. Consequently, different input traffic patterns and distributions were introduced into the model. The performance of the model is evaluated by analyzing its output. The design should collect a set of predefined measures that give a complete view of the impact of the paramters in the system, which comprise the traffic and scheduling algorithms in the context of this study.

This section provides an overview of the inputs and outputs of the AAPN model. The first part discusses the outputs required to evaluate the performance of the network. The second part presents input traffic options and their potential effect on the performance.

Performance Measures

Below is an outline of the performance measures collected by the simulation model:

End-to-End Delay :

Packets encounter different kinds of delays while travelling from their source to their destination. The total delay illustrates the time difference between the sending instant and the reception instant,and is referred to as End-to-End delay. Equation 3.1 illustrates its formulation.

$$\text{Total Delay} = \text{Propagation Delay} + \text{Transmission Delay} + \text{Queue latency} \quad (3.1)$$

Propagation Delay :

The time taken by the packets to travel though the media, optical fiber in this case, is referred to as propagation delay. It is calculated through the basic formula: Delay = Distance/Speed. While all packets travel at the same speed, distance is the variable in the equation. Thus the distance between the source node and the switch, and that between the switch and the destination sets the propagation delay of the packets. Table 3.1 presents delay values in different coverage scopes of all-photonic networks. Finally, it should be noted that propagation delays affect switching and scheduling time.

- Speed: Photons travel at a speed that equals tow thirds the speed of light, and is approximated to the speed of light in this study.
- Distances: Generally depends on the area coverage of the network, and specifically the location of the nodes in the network.

Table 3.1 Expected Delay values

Network coverage	Distance range (Km)	Bound	Delay (s)
LAN	1-10	Maximum	$0.5\text{x}10^{-4}$
MAN	10-100	Average	$2.4\text{x}10^{-4}$
WAN	≥ 100	Minimum	$4.5\text{x}10^{-4}$

Transmission Delay :

The time that elapses between sending the first and the last bits in a packet is called the transmission delay. This type of delay depends on the length of the packet and the bandwidth of the link. Transmission delays are independent of the switching and scheduling techniques. An edge node starts transmitting after it receives a grant from the scheduler, meaning that the transmission delay effect starts after scheduling and switching take place.

Queue Latency :

The controller in the core switch builds a matching matrix and sends signals to configure the switch at the beginning of every time slot. Packets stay idle in the queues during that scheduling and configuration time, that is referred to as queue latency. Queue latency contributes to the end-to-end delay of the packets and is totally dependent on the scheduling algorithm and the controller performance.

Scheduling time could form a bottleneck in a high speed network like AAPN. Moreover, OPNET is a DES[1] simulator that does not allow the study of the exact time spent by the scheduler. That however has been studied using another tool and will be discussed later. The simulation design devotes 1us for switching.

Loss Rate :

The AAPN design employs a cross-bar switch in its core. Cross-bar switches are non-blocking, which is crucial in meeting the performance requirements of the network. Consequently, the packet loss rate in the network is mainly due to the overflow of the queues, which includes queues storing packets and those storing left-requests. However,left over request buffers were made sufficiently large as to recover from losses in left over requests, which could lead to a system deadlock.

results showed that queues storing left-requests are less likely to overflow and so data queues are the main concern of this study.

That is to say, the rate by which packets are dropped in the network depends on the size of the VOQs, and the performance of the scheduling algorithm. Studying the effect of a scheduling algorithm on the rate of packet loss requires setting the size of the VOQs to a constant value. The VOQ size could cause a performance bottleneck in the network, so the effect of the size must form a fair tradeoff between the delay and the packet loss, this is further elaborated in the following section.

[1]Discrete Time Event

Utilization and Throughput :

The throughput of a network expresses the amount of data delivered from the source to the destination per unit time, and is measured by bits/s. Throughput is calculated by eliminating the amount of blocked traffic from the offered load, comprising the carried traffic in the network. Moreover, the blocking probability in the network is the ratio of dropped to offered packets. Finally the utilization of the network is the ratio of the carried traffic to the capacity of the links. Equation 3.2 illustrates the formulation of the link utilization measure in our design.

$$\text{Utilization} = {}^{\text{Carried traffic}}/_{\text{Capacity}}$$

$$\text{Carried traffic} = \text{Offered load}\,(1 - \text{BP}),\ \text{where BP is the Blocking Probability} \quad (3.2)$$

$$\text{BP} = {}^{\text{Dropped Packets}}/_{\text{Number of offered packets}}$$

Traffic Patterns and Distributions

The performance of a scheduling algorithm is highly affected by the nature of the input traffic and its distribution among network hosts, as final destinations. Traffic patterns are modeled by their arrival events. The study touches upon two types of arrival events; events that are independent and events that exhibit long range dependence among each other. The other aspect of traffic is its distribution among edges, it could be uniform or non-uniform.

Conventional network modelers treat traffic arrival events as being independent. Poisson arrival processes with exponential holding times form a convenient and easy approach for modeling such behavior. Other sophisticated models express the autocorrelation or short-range dependence (SRD) in bursty traffic. Such models are based on Markov-modulated Poisson or Bernoulli processes.

It was proven in the 1990s[28] that long-range dependence (LRD) is present in many types of networking traffic, including Ethernet LAN, WAN and ATM WAN traffic. Traffic streams exhibiting LRD are highly correlated at every timing scale and are so called Self Similar, refer to figure 3.1. Self-similarity is confirmed by examining the decay of the

autocovariance between traffic samples. The function decays exponentially in the case of SRD and hyperbolically in the case of LRD.

Figure 3.1 [28] illustrates self similarity in traffic, presenting packet counts collected from monitoring Ethernet traffic for 27 hours. The result is portrayed through five time scales, where the time resolution is increased from one plot to the next to show the autocorrelation between the samples. The plots demonstrate a similar pattern, distribution. That is, the traffic seems to exhibit the same behavior over long (minutes) and short (milliseconds) time scales.

There are levels of self-similarity in time series exhibiting long-range dependence. The Hurst parameter (H) is a measure of the level of self-similarity. Below is an illustration of the effect of the boundary values of H.

$0.5 < \mathrm{H} < 1$where :
$\mathrm{H} = 0.5$ indicates the absence of Self similarity, presenting Poisson traffic
$\mathrm{H} = 1$ indicates exact Self Similarity
$\mathrm{H} = 0.73$ real world traffic models

The effect of LRD on the utilization of the network is demonstrated in figure 3.2. The figure displays the utilization of the simulation model (refer to the following section for details about the model) employing three iterations of the iSlip scheduler under different levels of Hurst parameters, with 80% offered load. The graph corresponding to H=0.5 resembles the utilization under Poisson arrivals, which can be confirmed by the results presented later in chapter4. The graphs express the severity of the LRD effect on the network, where the graph representing highly self-similar traffic (H=0.9) shows a much lower utilization than that representing traffic with H=0.73.

The discovery of the self similarity nature of traffic raised doubts about modeling arrival events using conventional Poisson and Markov-modulated processes. Heavily tailed distributions are used to model self-similar traffic. The Pareto distribution is the simplest heavily tailed distribution that is hyperbolic over its entire range, refer to equation 3.3.

A random variable X has a heavy tail distribution if:

$$\Pr[\mathrm{X} > \mathrm{x}]\ x^{-\alpha} \text{ where } x \to \infty,\ 0 < \alpha < 2 \qquad (3.3)$$

$$\textit{Stearing the level of } Self - Similarity :\ \alpha = 3 - 2H$$

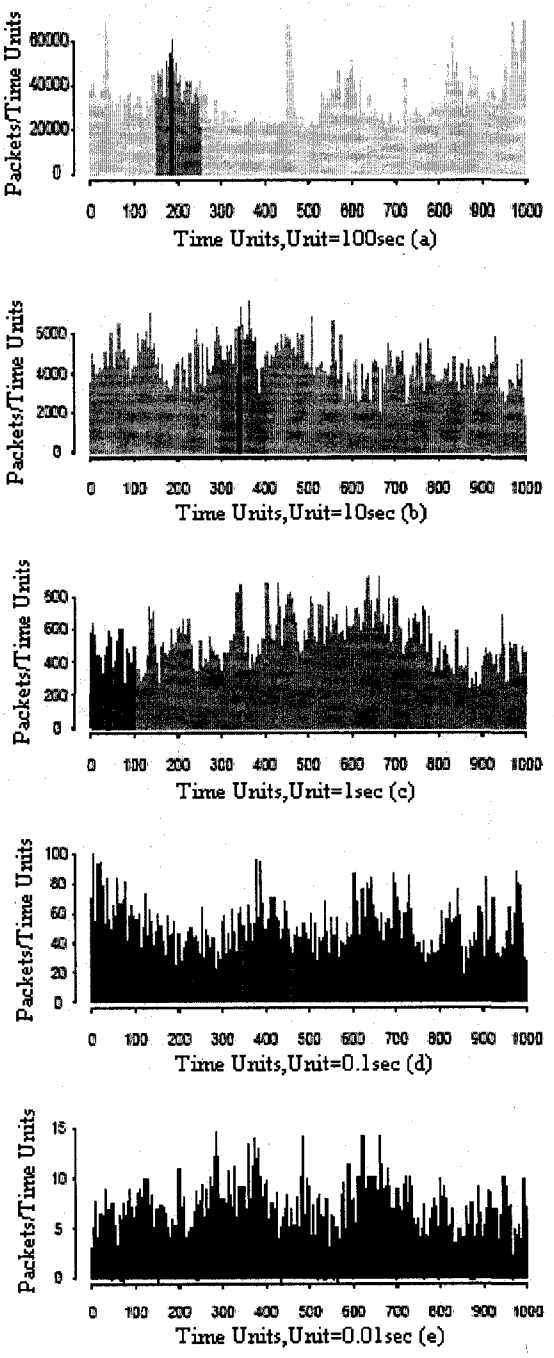


Fig. 3.1 Self Similar behavior of Ethernet traffic [28]

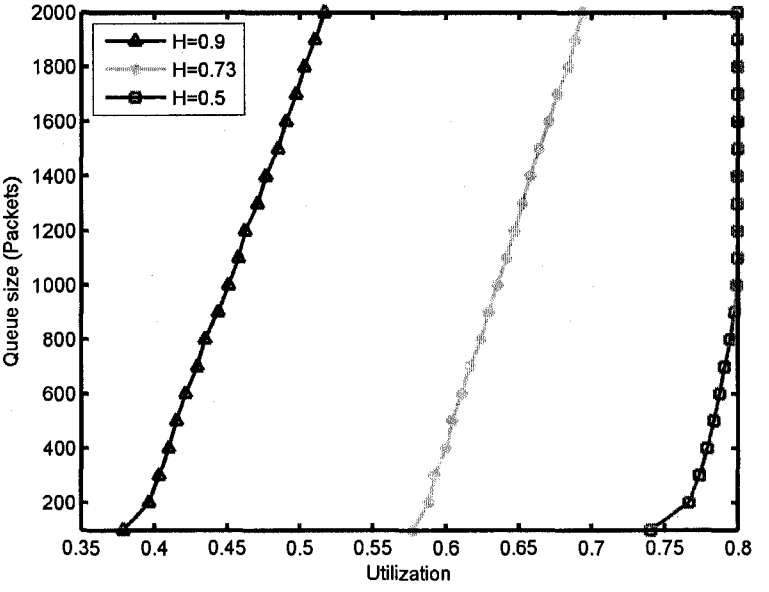


Fig. 3.2 The Effect of different Hurst parameter values on an AAPN MAN topology.

Self similar traffic could be modeled through one of the following approaches:

1. Generating packets with sizes drawn from a heavily tailed distribution.
2. Employing several independent and identical ON/OFF sources, where the period of traffic generation follows a heavy tail distribution. OPNET provided models generating self similar traffic based on this concept [29] [30]

The simulation design utilized in this study generates two kinds of traffic; independent arrival events and self-similar traffic. The independent arrival events are modeled using a Poisson process, with an exponentially distributed packet size. The self-similar traffic on the other hand is modeled through drawing the packet size from a Pareto distribution with H=0.73, the model was tested for H=0.5 to confirm the absence of Self-similarity and the validity of the approach.

As for the distribution of traffic among network hosts, two options were followed in the simulated design:

1. Uniform traffic: Generated traffic is uniformly distributed among destinations.

2. Non-uniform traffic: Traffic is distributed in a weighted fashion, where more traffic is sent to particular destinations than others. Following the approach used in [3],equation 3.5 is used to generate the non-uniform traffic, where λ_{ij} represents the traffic intensity from ingress i to egress j, refer to equation 3.4. The weight of distribution among the nodes is determined by w, where $0 \leq w \leq 1$.

$$\lambda_{ij} = \sum_{j=0}^{N-1} \lambda_{ij} = \lambda \left[w + (N-1)\tfrac{1-w}{N-1} \right] = \lambda = \sum_{i=0}^{N-1} \lambda_{ij} \qquad (3.4)$$

$$\lambda_{ij} = \begin{cases} 0 & if\ i = j \\ \lambda \left(w + \frac{1-w}{N-1} \right) & if\ j = (i+1) \bmod N \\ \lambda \left(\frac{1-w}{N-1} \right) & otherwise \end{cases} \qquad (3.5)$$

3.1.2 Simulation Design and previous work

An OPNET prototype was designed in previous work [3] to model the architecture of one layer of AAPN. The model was employed to simulate the performance of the network when PIM is employed as a scheduling algorithm. The reader is advised to refer to [27][3] for details about the actual implementation of the design in OPNET. This section briefly discusses an 8 edge node version of the model. The choice of performance shaping variables is explained in terms of the measures discussed in the previous section.

Simulation Design

An 8-edge node model of the design is illustrated in figure 3.3:

- Traffic sources: These modules are used to generate traffic according to a given distribution. The traffic is then offered to the ingress nodes.
- Links: Used to deliver packets and control signals (requests and grants) after a certain delay from an ingress node to the core switch, and from the core switch to an egress node. The delay in the links is set to a value that represents the network coverage.
- Edge nodes: These nodes form the ingress/egress points of the AAPN network. Traffic is passed from/to these nodes to/from the outside networks. Every edge node contains (N-1) VOQs for every other edge node in the network. For example edge node 1 in the figure, contains a VOQ for the following set of edge nodes {2, 3, 4, 5, 6, 7, 8}. When

an edge node receives a packet from a source, it stores the packet in its destination VOQ and submits a service request on its behalf in the next slot.

- Core Switch: This is the main module in the network. It contains a cross-bar that interconnects the edge nodes with each other. It also contains a controller that runs a scheduling algorithm to configure the cross-bar interconnections at the beginning of every time slot. The controller examines the service requests received from the edge nodes, does the matching and then configures the interconnections making paths for the packets to travel between the edge nodes.

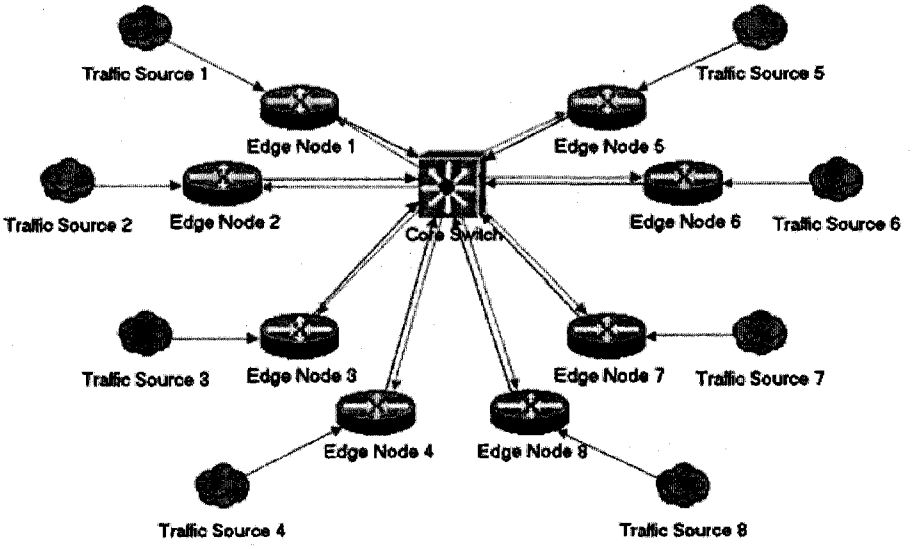


Fig. 3.3 A single layer of an 8x8 AAPN architecture model

Design Settings and Parameters

This section applies the concepts outlined in section 3.1 to the OPNET simulation model. The application is done through setting the design variables to control the process of examining the scheduling schemes.

AAPN Design settings :

Before discussing the choice of parameter settings, the following is a set of design parameters

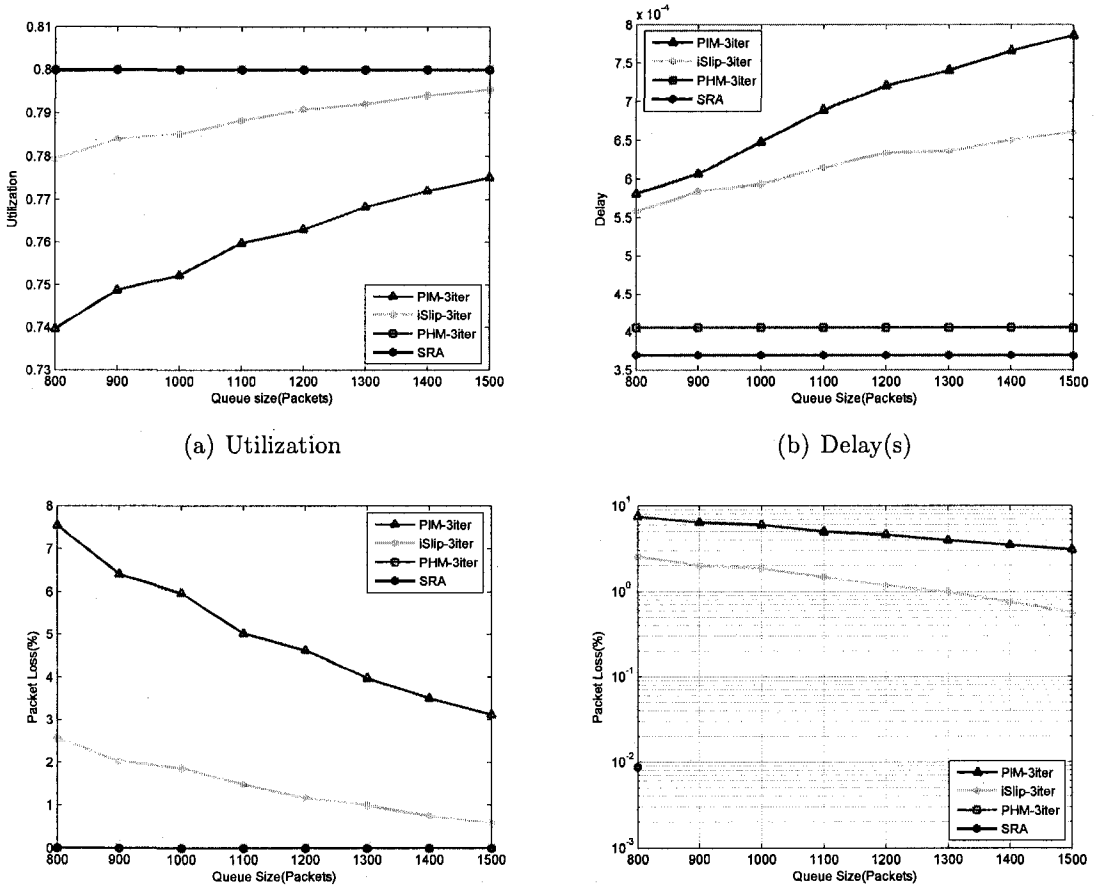
that are fixed by the architecture of AAPN:

- Link Capacity= 10Gbps
- Slot-Time= $10us$
- Slot-size= 10^5 bits this is a consequence of the Slot-Time and Link
- Switching time= $1us$ (guard band)

Variable Design Parameters :

Experiments and simulations are run to investigate the performance of a system. A system could have one or more variable parameters shaping its performance. Examining the effect of each of these variables requires setting the rest to constant values, variables that are set to constants are called control variables. There are four variable parameters in the AAPN model: The pattern of input traffic, Delay, VOQ size, and Scheduling algorithm. Since the scheduling algorithm is the variable under study, the rest become control variables. Control variables should be set to values that would leave a good margin for the experimental variable to influence the network.

- Traffic: Set to a single pattern and distribution at each simulation run.
  Two probability distributions were employed to generate traffic:
  1. Poisson arrivals: Packet size drawn from an exponential distributed with a mean of 1000bits
  2. Self-similar: Packet size drawn from a Pareto distribution with a mean of 1000bits The Hurst parameter, H=0.73, equation 3.3
- Delay: Dictated by the network coverage,as illustrated in table 3.1
- VOQ Size: The size of the buffers in a network has a direct prominent effect on the amount of dropped packets and the queue-latency delay, which indirectly affects other network performance measures like the utilization of the links. It was mentioned previously that AAPN is based on best effort service, making the choice of the VOQ sizes very critical. In other words, to be able to support applications with stringent performance constraints, the VOQ size should not form a bottleneck in the design. Taking Voice over IP as an example, the service is required to support a delay less than 70ms and a packet loss of less than 0.1%. The size of the VOQ is selected by collecting performance measures from each of the schedulers upon sweeping the VOQ size from 800 packets to 1500 packets. The measures were collected for MAN and WAN topologies



(a) Utilization

(b) Delay(s)

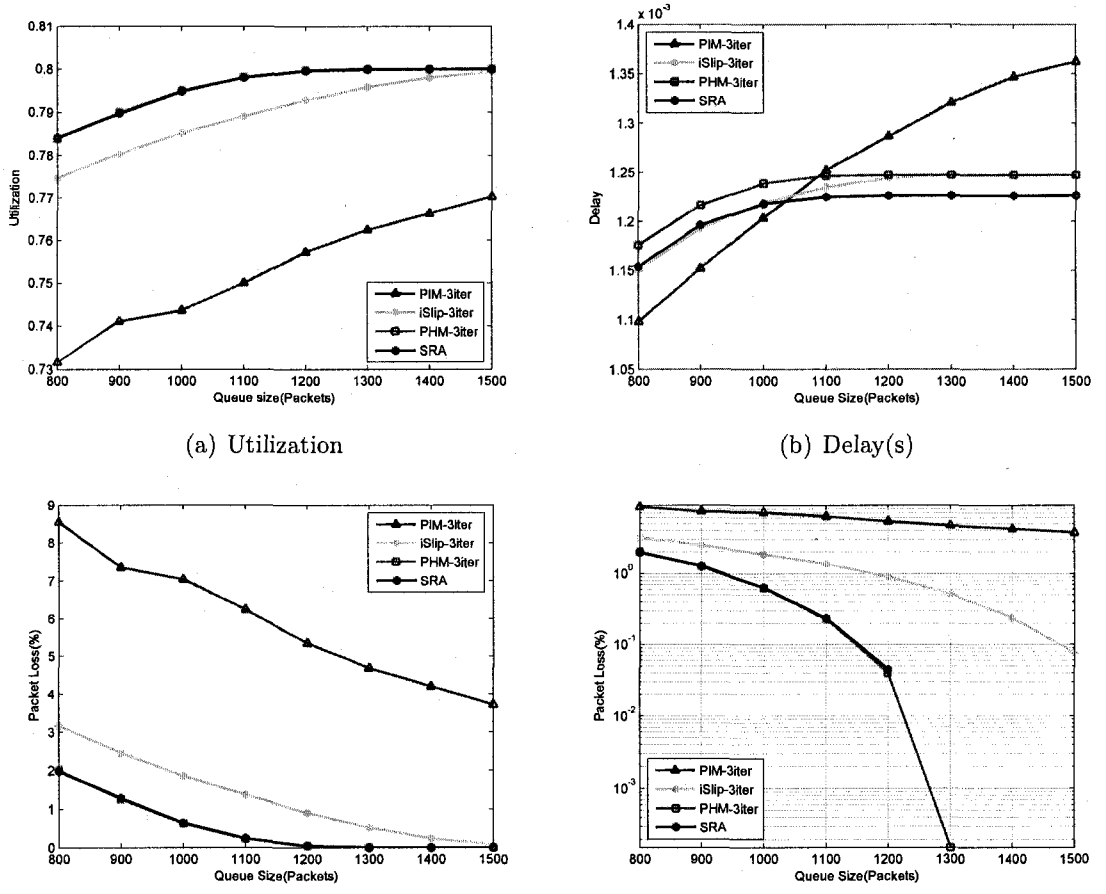
(c) Packet Loss, linear scale

(d) Packet Loss, Log scale

Fig. 3.4 Performance measures with respect to VOQ sizes, MAN topology and 80% offered load

with an 80% offered load. The plots are presented in figure 3.4 and 3.5. One should note that the PIM and SRA plots coincide in figures 3.4(a) and (c). Moreover, figure 3.4(d) demonstrates the fact that PIM and SRA do not cause any loss for that range of queue size, as they do not appear in the figure.

Both figures demonstrate the tradeoff between the delay of the packets (b) versus the packet loss (c and d) and the utilization of the links (a). As the queue size increase, the delay of the packets increase which is unfavorable. On the other hand, the utilization



(a) Utilization

(b) Delay(s)

(c) Packet Loss, linear scale

(d) Packet Loss, Log scale

Fig. 3.5 Performance measures with respect to VOQ sizes, WAN topology and 80% offered load

and the packet loss rate favor such an increase in the queue size. In setting the size of the VOQs, one notes the most stringent constraints to be supported by the network, and makes a fair trade off between the measures. For example figure 3.4 shows that to support Voice over IP, the minimum queue size should be 1200 packets, that is to meet the packet loss requirement. Moreover,the delay would also be supported since it is a MAN topology.

3.2 Hardware Simulation iSlip

3.2.1 Choice of hardware: FPGA over ASIC

3.2.2 Design

The study of the hardware implementation of iSlip did not get as much attention as the study of its performance. Serpanos et al. discussed the basic design of the request-grant-accept protocol in hardware, and provided examples about the requirements for the FIRM scheduler in [12] [16]. Another study by Gupta and McKeown [13] proposed a pipelined implementation of iSlip, where the scheduler overlaps the accept phase of one iteration with the request-grant of another, saving clock cycles in an iterative run of the algorithm. The delay imposed by the arbiters in iSlip has a significant effect on the speed of the scheduler, thus the design of fast arbiters is very critical. Reference [13] discussed several arbiters' implementations, presenting the tradeoffs between the hardware requirements and complexity of each design. The simulation model used in this study adopts a simplified model that improves the design in [12] and utilizes the optimal arbiter design proposed in [13].

iSlip is a distributed scheduler that makes decisions through the handshaking protocol described in chapter 2. The protocol was explained in detail and is based on three stages: Request, Grant and Accept. The model employed for this study merges the Request and Grant phases into a single phase, which enhances the complexity of the design without altering the functionality of the system. Figure 3.7 shows how the requests can be forwarded directly to the Grant blocks without passing through intermediate blocks. However,each stage receives a set of control signals, processes the signals and generates a set of subsequent signals that steer the functionality of the following stage, figure 3.7. Each phase is realized by N identical blocks operating in parallel. An additional block is required to update the pointers in the Grant and Accept blocks. Furthermore, a three-state finite state machine (FSM) is utilized to trigger the appropriate block at each phase of the protocol. The two phases of an iteration take effect in one clock cycle. One should note that merging the Request and the Grant phases only involves the hardware units; the data flow however takes place in sequence.

3.2.3 Block Functionality

1. Grant Blocks:
   Grant blocks take request signals and arbiters' positions as input. The blocks utilize arbiters to generate grants (their output control signals). Granting signals are passed to Accept blocks and the arbiter updating block.
2. Accept Blocks:
   Accept blocks take grant signals and arbiter's positions as input. Just like grant blocks, they utilize arbiters to choose an output from the set of granting outputs. If the scheduler is running the final iteration, the output control signals of this block configure the switch for that time slot, otherwise the signals are used to block the matched inputs/outputs from being considered in the following matching iteration.
3. Arbiters Updating Block:
   The functionality of this block is dependent on the scheduler. The discipline by which arbiters are updated differentiates between arbitration schemes. In other words, this block is the only block that should be changed when implementing different arbitration schedulers. The block controls the location pointed at by the arbiter after a certain stage, corresponding to the output of that stage. Only one arbiter updating block is utilized in this model. The block updates the grant arbiters while the system is in the Accept phase, and updates the accept arbiters while the system is in the Grant phase. The appropriate functionality of the block is triggered by the FSM, while the values to be-updated are fed by the outputs of the Grants and Accepts blocks.
4. Controller unit:
   The controller synchronizes the operation of the blocks to accomplish an ordered execution within a single iteration run. The controller utilizes a three-state FSM, which generates control signals that enable/disable each of the blocks. Additionally it sets the mode of operation of the arbiters updating block to Grant or Accept.

3.2.4 Arbiter units utilized in the Grant and Accept blocks

The functionality of the arbiters was discussed in chapter 2. The model utilizes the design discussed in [13], which proved to be optimal. The arbiters were implemented using two Priority encoders (smpl_PE), a thermo decoder (discussed below) and some logic gates used for selection, refer to figure 3.6 below:

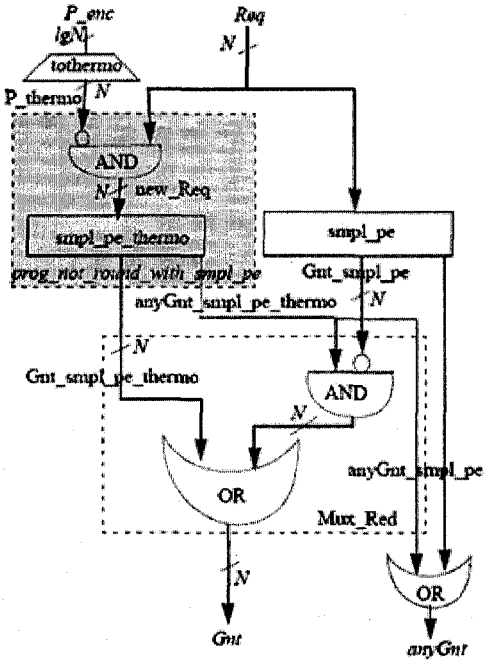


Fig. 3.6 Hardware design of round-robin arbiters, proposed in [13]

Signals:

Input:

1. P_enc: The current position pointed at by the arbiter, $\log_2 N$ bits
2. Req:

   Grant blocks: Input requests ,N bits

   Accept blocks: Output grants,N bits

Output:

1. Gnt: Results of the arbitration process, N bits
2. anyGnt: Signals the presence of a grant

The figure shows that the Req from all the inputs takes two different paths. The first path is the one that indicates the presence of a request from the inputs (i in Req [i]) between P_enc and (n-1). The other path indicates the presence of requests from the inputs

between 0 and (P_enc - 1). Finally a decision of which request is granted takes place at the final block (Mux_Red).

Block-level explanation of the proposed arbiters' design

1. Priority Encoders:
   The smpl_pe_thermo and the smpl_pe are Priority Encoders (PE), their functionality is demonstrated by the truth table 3.2, assuming N=4

   Table 3.2 Priority Encoder

Req(3)	Req (2)	Req (1)	Req (0)	Y(1)	Y(0)	C
0	0	0	0	X	X	0
X	X	X	1	0	0	1
X	X	1	0	0	1	1
X	1	0	0	1	0	1
1	0	0	0	1	1	1

   Y is output and C is a variable used to indicate the case when none of the inputs send a request. In the context of the proposed design, C indicates that there is no requests from any of the inputs that lie between the P_enc to (n-1). C is used as a selector for the Mux_red block.

2. Thermo-Decoder:
   As described in [13] the thermo_decoder is used to decode the $\log_2 N$ bit P_enc input into a four bit value, to simplify the remaining functionality (more elaborated in the following sub-section). The following truth table demonstrates the functionality of the decoder:

   Table 3.3 Thermo_ Encoder

P_enc(1)	P_enc (1)	Y(3)	Y(2)	Y(1)	Y(0)
0	0	0	0	0	0
0	1	0	0	0	1
1	0	0	0	1	1
1	1	0	1	1	1

3. Prog_not_round_with_smpl_PE:
   The block exploits the functionality of a thermo_decoder in conjunction with a negator and an AND gate to filter out the requests from input 0 to (P_enc - 1). The remaining

requests are then fed to the priority encoder to choose among them.

4. Mux_red:
   The block acts as a selector that chooses among the decisions made by each path using the anyGnt_smpl_pe_thermo

Example:

Input:
Req="1101"
P_enc="01"
Intermediate:
P_thermo="0001" ⇒ (NOT)P_thermo="1110" allowing everything beyond the pointer to pass.
new_Req="1100"
new_Req="1100"
Output:
Gnt_smpl_pe_thermo="0100"

3.2.5 Implementation

The design was implemented in Quartus II 6.1 [31], a hardware CAD tool. Functional and timing simulations were run, and these confirmed the desired functionality of the design and determine the propagation delays expected in the circuit, respectively. Optimization techniques can also be deployed to get the best design for a given technology.

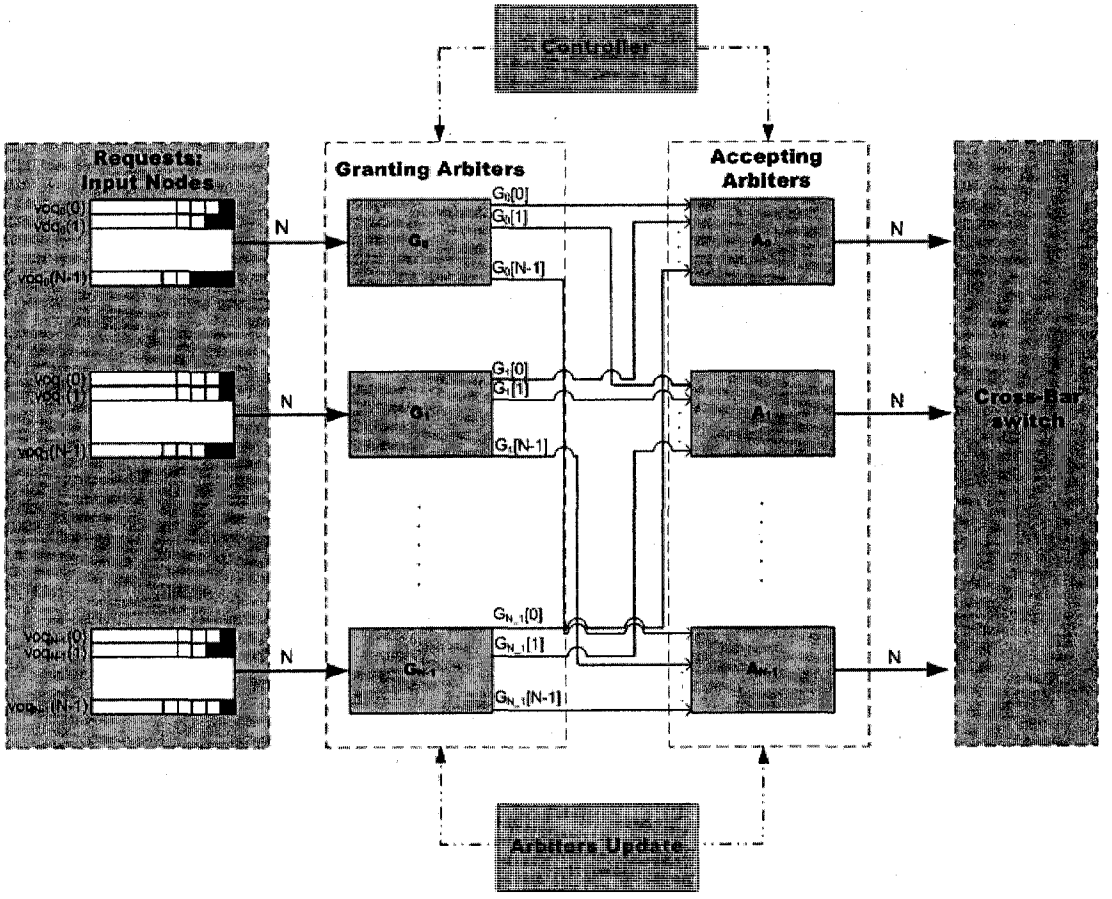


Fig. 3.7 iSlip Hardware simulation model

Chapter 4

Results and Analysis

Four Matching algorithms, PIM, iSlip, PHM and Adapted SRA were discussed in section 2.3. A simulation model emulating the AAPN design was discussed in chapter 3. The first part of this chapter integrates those sections, presenting results obtained from coding the schedulers in the simulation model. The model was simulated for an 8-edge node environment under the control of each of the schedulers. Different input traffic patterns were employed to evaluate the model's ability to adapt to various scenarios. The results reported in this section were all generated on average basis[1]. The literature reported performance results and properties about each of the schedulers. Reassessing these results forms the starting point of our analysis. The modified schedulers were then simulated in application to the AAPN model. The first step utilized a conventional Poisson arrival process and uniform destination distribution, structuring a comparison basis for the analysis. Afterwards, the effect of more realistic models, with a combination of traffic distributions and arrival behaviors are tested. The second part of the chapter presents results obtained from the hardware simulation model. An association between these results and others, reported in the literature, is provided to compare the hardware complexity of the algorithms.

[1]Results were generated from each of the nodes, and the average of these results was noted

4.1 Performance Results

4.1.1 Basic schedulers

Performance of a single iteration of each of schedulers

Figure 4.1 demonstrates the performance of a single iteration of each of the scheduler's algorithms under a Poisson uniformly distributed traffic in MAN topology. figure 4.1(a) confirms that a single iteration run of PIM achieves a maximum link utilization of 63%, even when the network is fully loaded. iSlip does not reach the reported 100% utilization at 100% load, which could be due to more than one reason; like synchronization among the pointers, VOQ size and the fact that networks start behaving strangely when fully loaded. PHM shows superiority among PIM and iSlip in all four measures. Furthermore, the performance of the maximum size match (SRA) is very close to that of PHM.

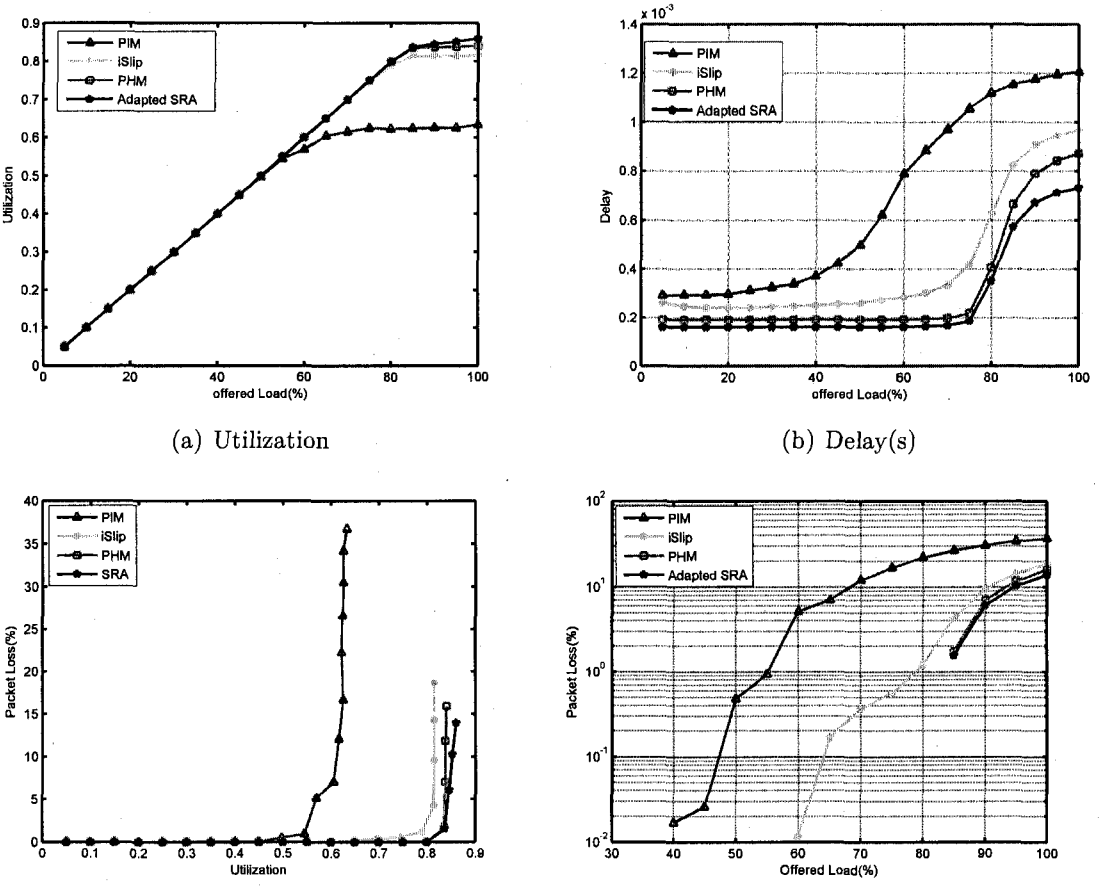
The figure illustrates the following facts:

1. The performance of one iteration of PIM is unacceptable
2. The performances of PHM and the adapted SRA are very similar, while it was proven that SRA is very complex in comparison. The slight improvement in performance can be traded for the sake of complexity.

From this point on, the adapted SRA is replaced by PHM and will not be reported in the following performance measures.

Convergence speed of the schedulers

Figure 4.2 demonstrates the convergence speed of each of the three algorithms, under Poisson, uniformly distributed traffic load of 60% in MAN topology. The simulations show that PIM exhibit a speed of convergence of 6 iterations for an 8-edge node network; that is of the order of $O(\log_2 N)$, as claimed in [4]. However a rough evaluation would confirm the complexity of PIM reported in the literature. iSlip on the other hand shows the speed of convergence of 2 iterations. Lastly, PHM confirms the reported facts, stating that the algorithm converges in $O(1)$ time.



(a) Utilization

(b) Delay(s)

(c) Packet Loss, linear scale

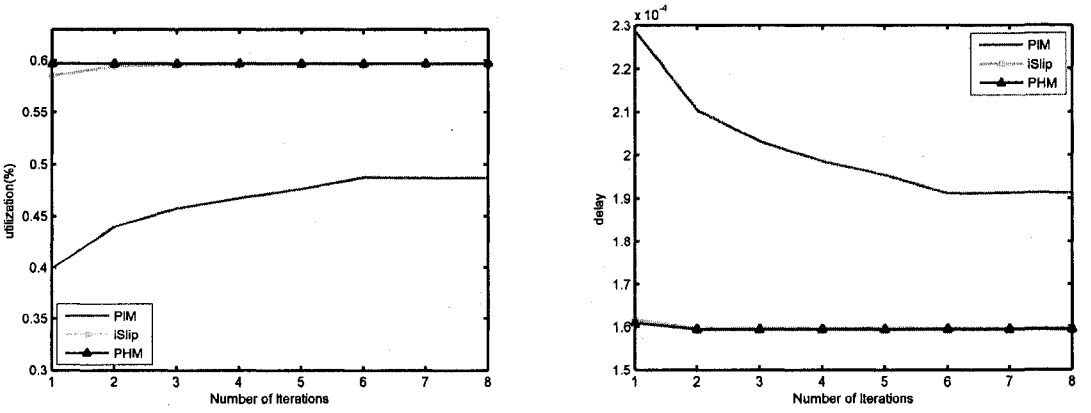
(d) Packet Loss, linear scale

Fig. 4.1 1-iteration run of each of the scheduling algorithms, MAN topology with Qsize=1000packets

4.1.2 Performance measures of the adapted schedulers with different input traffic

Poisson arrival process, with traffic uniformly distributed among network hosts

Figure 4.3 demonstrates the performance of the network under a conventional traffic model. The arrival process follows Poisson distribution and the traffic is uniformly distributed among the edges; w=0 in equation 3.4. The model is simulated for a MAN topology and three iteration runs, for each of the scheduling algorithms. The results demonstrated in figure 4.3 form a comparison base, since they were generated from idealistic traffic.

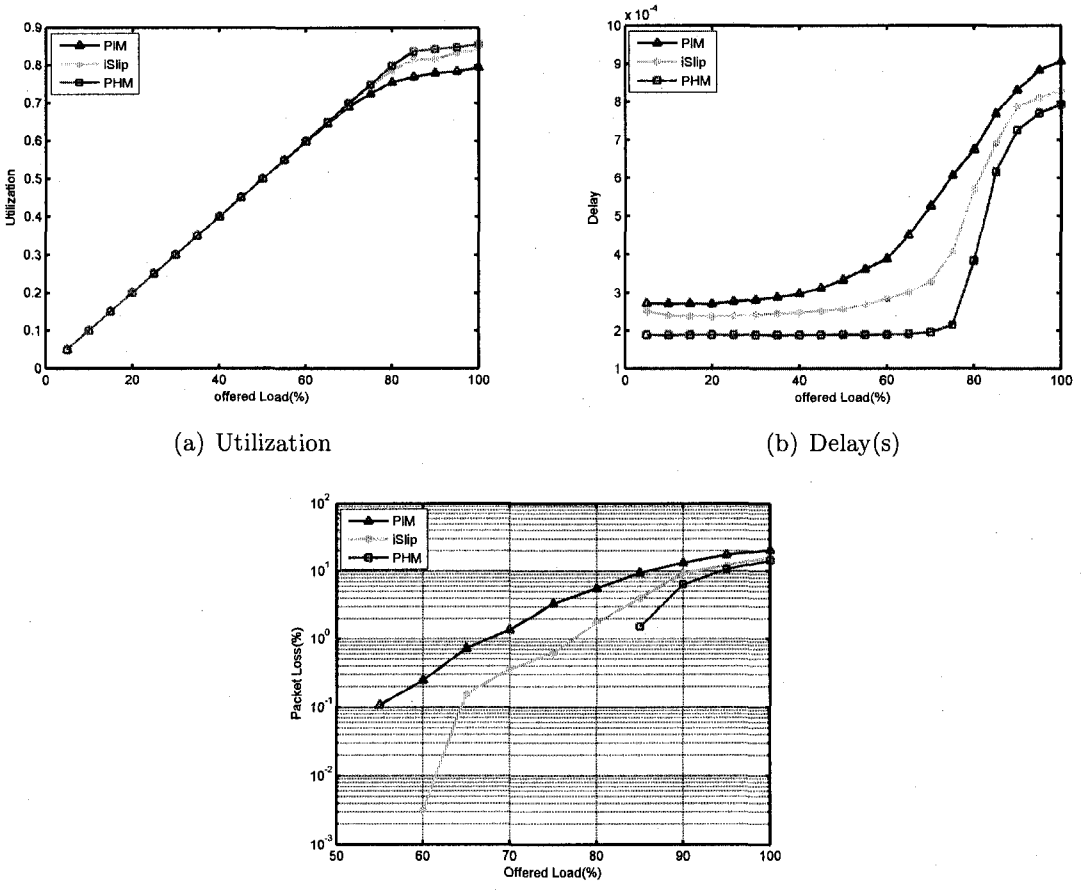


(a) Utilization (b) Delay(s)

Fig. 4.2 Convergence of the algorithms

Poisson arrival process, with traffic non-uniformly distributed among network hosts in a MAN topology

The results presented in figure 4.4 demonstrate the performance of a more realistic network model, where the arrival process is still Poisson but the traffic is not uniformly distributed, w=0.3 in equation 3.4. Comparing the graphs in figure 4.4 with the standard results in figure 4.3, one would note that the performance difference corresponding to traffic loads between 40%-80% is insignificant. The throughput of the network drops when the traffic is non-uniformly distributed, which is expected. However figure 4.4(a) shows that the throughput under PHM and iSlip increases linearly as compared to the throughput behavior in the standard case (figure 4.3(a)). In the case of PIM, the throughput stabilizes to a constant value at about 85% load, which enforces the limitation of the random arbitration in PIM as opposed to other the schedulers. The amount of packet loss in low traffic loads of non-uniformly distributed traffic is significantly higher than that in uniformly distributed traffic (figure 4.4c). Figure 4.3c indicates that packet loss is very low for traffic loads below 55%, whereas figure 4.4b indicates that traffic loss starts from a 20%-30% load. The delay performance graphs in figure 4.4(b) demonstrate the superiority of iSlip and PHM over PIM. One would notice a peculiar behavior around 30% load, where the graphs reach a maximal point and then ramp down again, with the exception of PIM that keeps on increasing. The peak is due to the non-uniform distribution of the traffic. Some VOQs



(a) Utilization

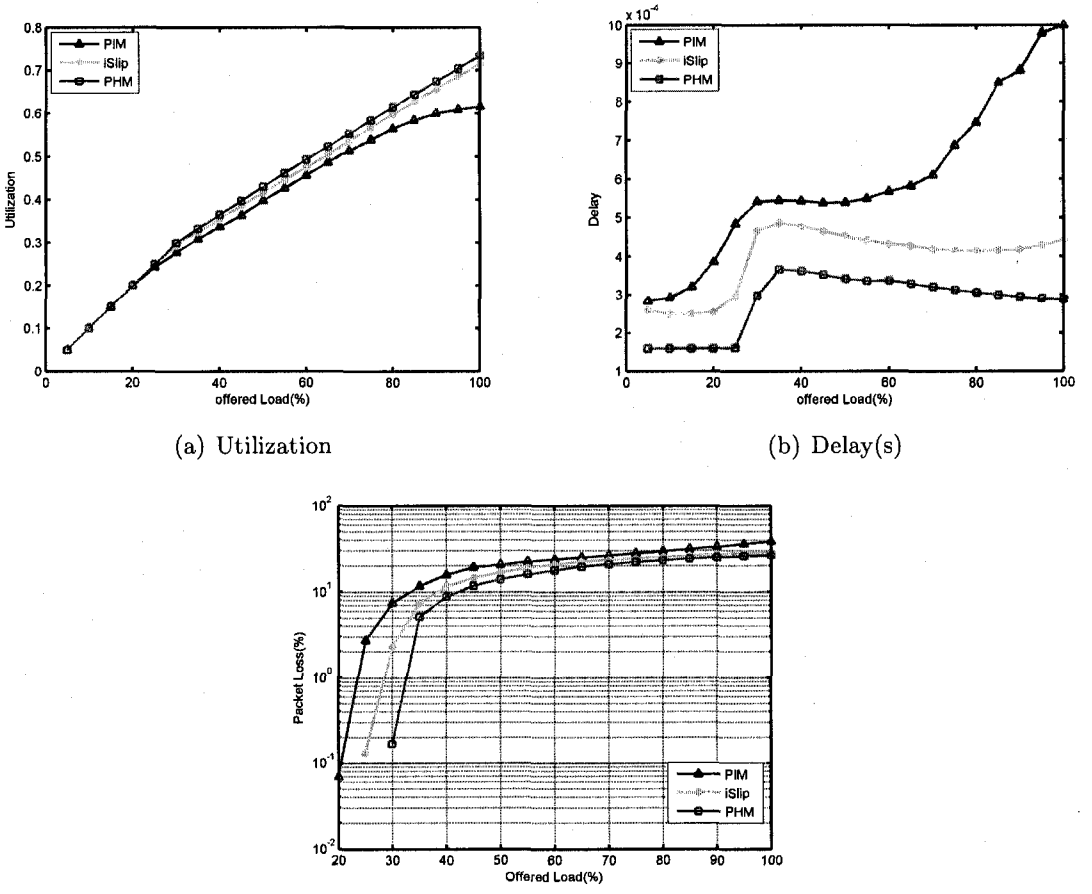
(b) Delay(s)

(c) Packet Loss

Fig. 4.3 Network performance under Poisson, uniformly distributed traffic, MAN topology, VOQ size=1000 packets, utilizing 3-iterations runs

get blocked and start losing packets while keeping the delay of the queued packets the same [2], whereas other VOQs queue more packets, ones that will experience larger delays. Averaging the delay over such non-uniform distribution produces higher values than the regular case where all nodes block packets, which starts when the percentage of packet loss becomes more pronounced (around 30% load). PIM exhibits an increasing delay, one that is higher than that reported in figure 4.3(b). Whereas, the network delay imposed by employing iSlip and PHM stabilizes after a 50% load.

[2]Packet loss and delay are independent measures



(a) Utilization

(b) Delay(s)

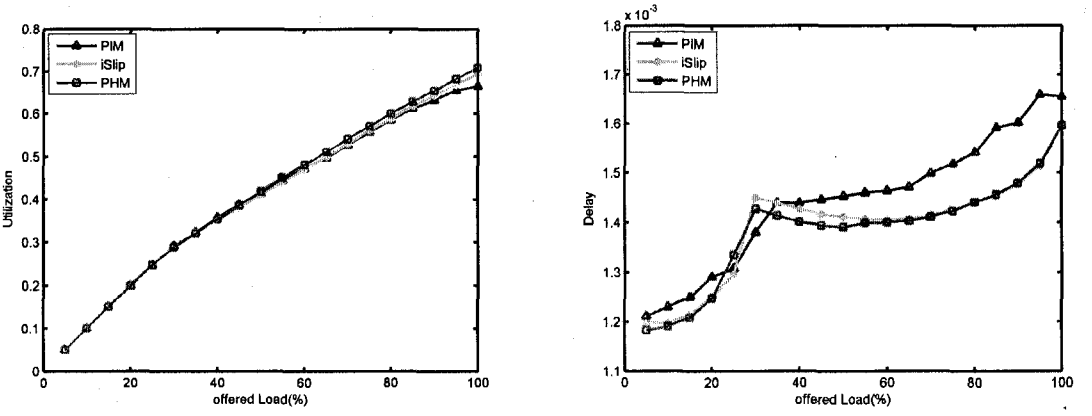
(c) Packet Loss

Fig. 4.4 Network performance under Poisson, non-uniformly distributed traffic (w=0.3), MAN topology, VOQ size=1000 packets, utilizing 3-iteration runs

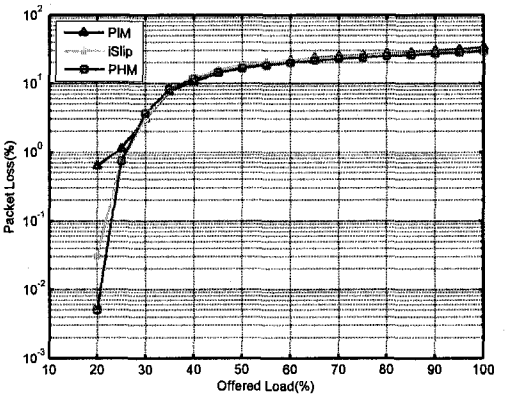
Poisson arrival process, with traffic non-uniformly distributed among network hosts in a WAN topology

Figure 4.5 demonstrates a similar effect on the network performance, as the previous simulation, but in a WAN topology instead of MAN. The graphs demonstrate higher delays and packet loss percentages. The delay performance in figure 4.5(b) is somewhat different than that displayed in the MAN topology. The figure shows the same peak at 30% load, but the behavior of the graphs before and after that peak is different than that in figure 4.4 (b).

The graphs illustrate a vigorous increase in the delay under traffic loads of 20%-25%, that is when packet loss starts in few nodes (figure 4.5(c)). For higher loads, the percentage of packet loss increases, more nodes start blocking packets, and so the delay of each of the graphs increases as expected.



(a) Utilization (b) Delay(s)



(c) Packet Loss

Fig. 4.5 Network performance under Poisson, non-uniformly distributed traffic (w=0.3), WAN topology (200Km), VOQ size=1000 packets, utilizing 3-iteration runs

Self Similar traffic, uniformly distributed among network hosts in MAN topology

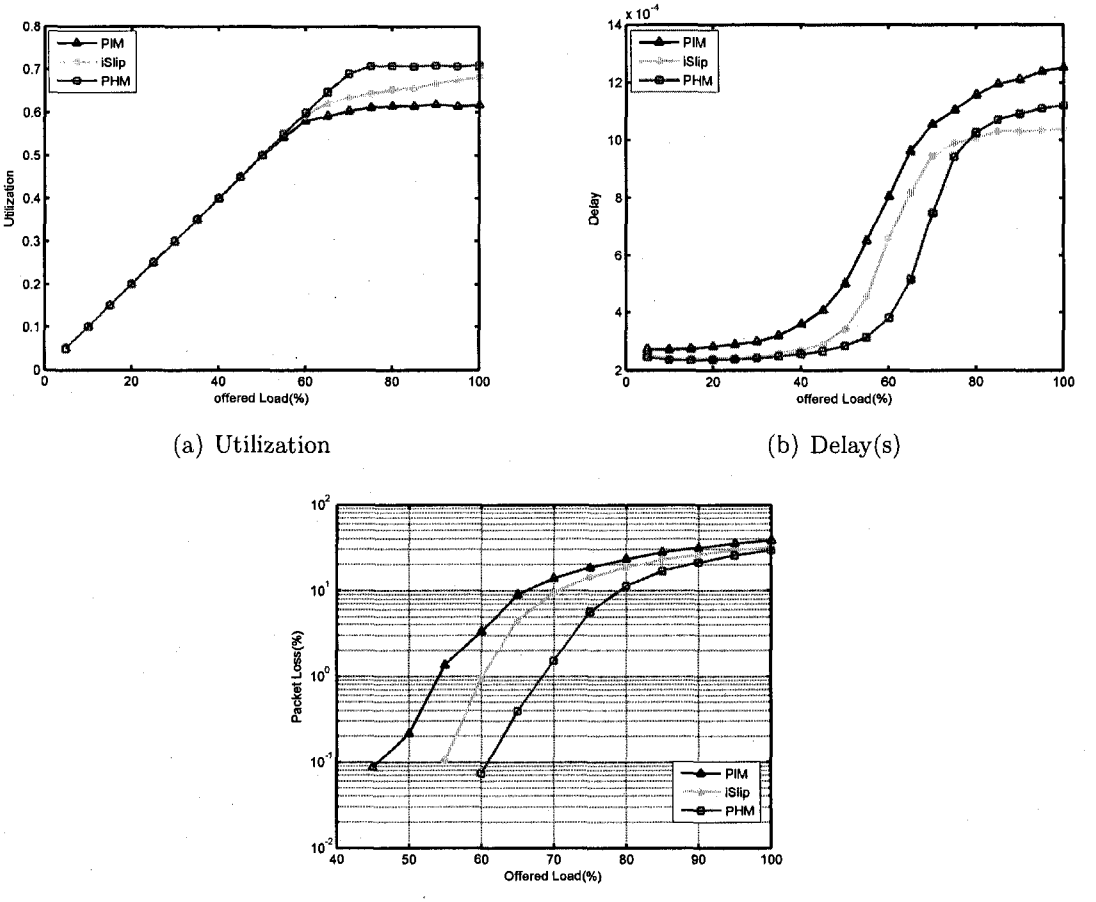
Figure 4.6 shows a realistic model, where traffic samples exhibit LRD. The model is simulated for three iteration runs of each of the scheduler algorithms in a MAN topology. Generally the behavior of the each of graphs is similar to its equivalent in figure 4.3, indicating that self similar traffic would be supported in the AAPN model. The network throughput drops under Self similar traffic, as compared to Poisson arrivals. PIM achieves very poor utilization, figure 4.6(a). The increase in packet loss vs offered load is much steeper in the case of Self similar traffic versus Poisson, but stabilize at about the same load, 80%. It should be noted that for Poisson traffic loads lower than 80%, the packet loss is insignificant upon utilizing PHM, refer to figure 4.3. However packet loss in Self similar traffic starts from a load of 60%. Moreover, the delay of the network under PHM is greater than that imposed by iSlip for traffic loads greater than 70%, which is not the case for Poisson traffic. One would conclude that PHM would still meet the stringent performance requirements of a MAN network under Self similar traffic, as long as the load is kept below 75%.

Self Similar traffic, uniformly distributed among network hosts in a WAN topology

Figure 4.7 demonstrates the same effect on the network performance, as the previous simulation, but in a WAN topology instead of MAN. The graphs demonstrate higher delays and packet loss percentages. One would conclude that the network performance in this case would support applications with stringent requirements, only if the traffic load is kept below 60%, which would achieve a throughput less than 0.57 in the best case scenario, when PHM is employed.

Self Similar traffic, non-uniformly distributed among network hosts in a WAN topology

Figure 4.8 indicates that network with self similar, non-uniformly distributed traffic load has a poor performance under the employment of each of three schedulers. The behavior of the graphs in Figure 4.8(b) is comparable to the behavior of the graphs in figure 4.5(b).



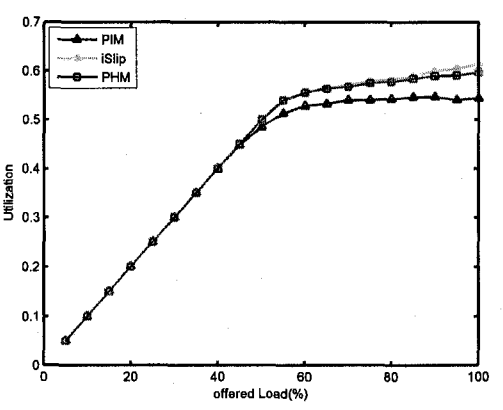
(a) Utilization

(b) Delay(s)

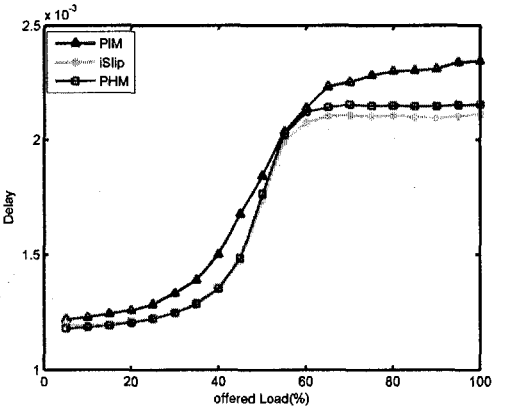
(c) Packet Loss

Fig. 4.6 Network performance under Self Similar, uniformly distributed traffic, MAN topology, VOQ size=1000 packets, utilizing 3-iteration runs

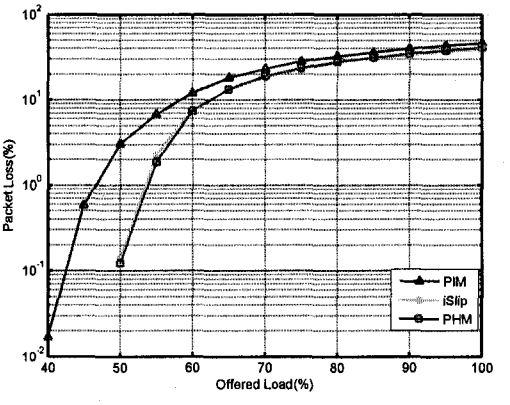
The percentage of packet loss in figure 4.8(c) demonstrates a sudden increase at 80% load, which leads to saturation of the utilization graphs and a steep increase in the delay graphs.



(a) Utilization

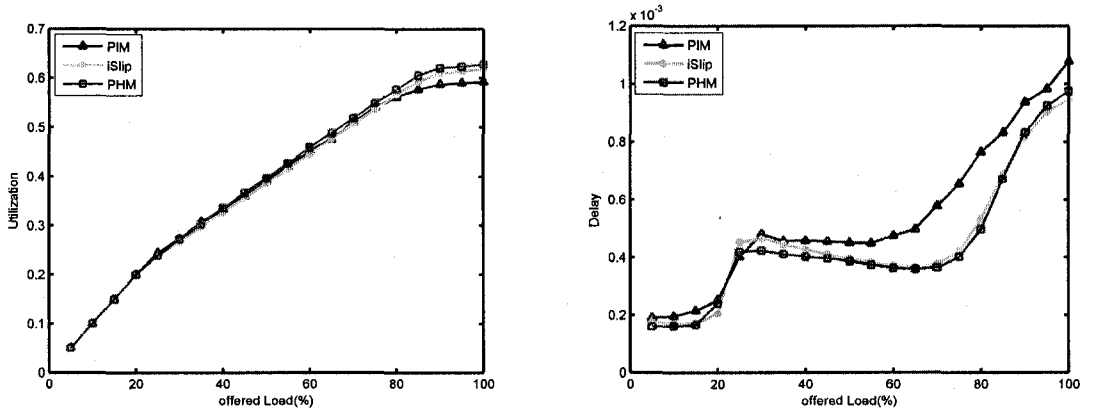


(b) Delay(s)



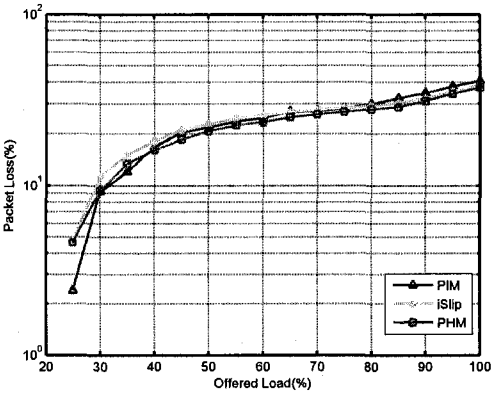
(c) Packet Loss

Fig. 4.7 Network performance under Self Similar, uniformly distributed traffic, WAN topology, VOQ size=1000 packets, utilizing 3-iteration runs



(a) Utilization

(b) Delay(s)



(c) Packet Loss

Fig. 4.8 Network performance under Self Similar, non-uniformly distributed traffic, WAN topology, VOQ size=1000 packets, utilizing 3-iteration runs

4.2 Hardware complexity results

This section presents the results collected from the hardware implementation of iSlip on a Cyclone II, EP2C70F89618 device [31]. The results agree with the research findings reported in the literature, considering the lack of protocol-steps pipelining in our design. The main concern of the study is to evaluate the worst timing requirement of iSlip, and when that was met, as table 4.1 confirms, there was no need to update any further. Moreover, PIM was never implemented in hardware due to its hardware complexity that requires large running times, failing to meet AAPN's $10us$ timing costraint. Whereas, many studies showed that PHM is much simpler to implement and requires less running time.

The results demonstrated in tables 4.1 and 4.2 were collected from running the design in a 4x4,8x8 and 16x16 network environments, to investigate the scalability of the design. It should be noted that the simulation device has a significant impact on these results. The layout of a device sets a lower limit to the propagation delay of the signals, where signals are sent from one unit to another, that contributes to the difference between the results obtained in our design and other designs.

4.2.1 Timing Requirements

Table 4.1 illustrates the results obtained from running the iSlip implementation into the Cyclone II device. The clock frequency in the table is set to a value that would support the delay of the bottleneck unit in the design, the Grant block. That is, before running the operation of the whole design, a timing analysis tool was employed to determine the bottleneck and its timing requirement. Moreover, the design accounts for other timing requirements, such as the time needed for the circuits to stabilize. One would note that the clock frequency decreases as the network expands, since more nodes require more logic, more memory and higher processing times.

The table confirms the applicability of iSlip in AAPN up to 16 nodes. Moreover, the Tiny Tera project [13] confirmed the applicability of a pipelined version of the algorithm, the results reported a running time of 51 ns for three iterations of iSlip in a 32x32 environment, on a Xilinx device. The results from that study prove the significance of pipelining on the running time of the algorithm.

Table 4.1 Timing Results

N	Worst-case propagation delay (ns)	Clock Frequency MHz	Total Time iterations	time (ns)
4	8	100	2	40
			3	80
8	13	66	3	80
16	22	45	3	130
			4	180

4.2.2 Resource utilization

Table 4.2 outlines the hardware requirements of iSlip in the Cyclone device. The requirements obviously change from one device to another, but the results give a rough idea about the general requirements of the implementation. Moreover, it should be noted that the EP2C70F89618 device would not support a 32x32 implementation of the design, while other industrial devices would.

Table 4.2 Resource Utilization Results

N	Total logic elements	Total combinational functions	Dedicated logic registers
4	265	249	96
8	1495	1495	374
16	8777	8477	1432

The timing constraints of PHM were studied in [11] and [15]. An analytical approach was followed by simulations on an ASIC library, lca300k.alf in [11]. The results confirmed that PHM has a running timing that is much shorter than that in other arbitration based algorithms, such as RDSRR. [15] on the other hand, confirmed the applicability of PHM to such high speed networks by synthesizing the algorithm on several devices. Comparing the results obtained by our design and those reported in reference [15], indicates that generally PHM is faster than iSlip, but requires more logic units.

Chapter 5

Conclusions and Future work

The study of time-slotted schedulers in AAPN is presented in this thesis. Four algorithms were researched and employed in the study. The performance of the network under each of the proposed schedulers was tested using an OPNET model, one that emulates the infrastructure of AAPN. Furthermore, the hardware implementation of the algorithms, which concerns hardware requirements and running speed, is evaluated to establish a full view of their requirements and compare them to the constraints set by AAPN. This chapter provides a summary of the outcomes of the study, and suggests research areas to be further exploited for the scheduling process in AAPN.

5.1 Summary and Conclusion

The first and most obvious conclusion of the study is the fact that PIM does not meet the criteria set for AAPN. The performance of the network, under PIM, fails to meet the minimum requirements unless more than $\log_2 N$ iterations are run, that on the other hand consumes so much time, and fails to meet the $10us$ timing constraint. However, one would argue that features of the Adapted-PIM [3] would conform better to AAPN. While it was confirmed that Adapted-PIM enhances the performance of the network, the algorithm was not investigated in terms of its hardware complexity. Adapted-PIM requires extra control over the network queues, to monitor their capacity, which adds to the complexity of PIM, making its implementation impractical.

SRA was disregarded from the simulations due to the fact that PHM was able to deliver comparable performance results with much lower complexity.

The performance of the network under PHM is better than that under iSlip. Both algorithms do not perform so well under non-uniform traffic, load-balancing would resolve that issue while adding on the control overhead, and so the complexity of the schedulers.

The effect of self similar traffic on the network is acceptable for some applications in MAN topologies. However, the performance would not be acceptable for applications in WAN topologies.

The study revealed that the AAPN scheduler could employ iSlip for a network containing up to 64 nodes, while meeting the $10\mu s$ timing constraint. Furthermore, the literature illustrated several studies, utilizing different ASICs and FPGAs, where the implementation of PHM appeared to be more practical than arbitration algorithms.

Lastly it is worth mentioning that PHM and its variations could be utilized in implementing QoS in AAPN. Such an application, along side the algorithms' high running speed and high performance, demonstrate the superiority of PHM among the rest of the algorithms studied in this thesis.

5.2 Future work

5.2.1 Performance

Load balancing techniques have been adopted in hybrid-networks to mitigate the effect of non-uniform traffic. The application of such techniques to AAPN would definitely enhance the performance.

Self-similar traffic is inevitable, and performance measures proved that it is hardly supported in MAN topologies in AAPN. Employing a hybrid of distributed scheduling approaches could resolve that matter. The determination of the level of self-similarity of traffic in the edge nodes, followed by an adaptive scheduling approach could achieve better performance measures, while increasing the complexity.

5.2.2 Hardware Measures

The lack of protocol pipelining in the hardware implementation of iSlip is considered a limitation. While the minimum timing requirements were met, pipelining would support networks larger than 64 nodes.

The hardware implementation of PHM is most important step in continuing the

work done in this thesis.

References

[1] http://www.aapn.mcgill.ca.

[2] X. Liu, A. Vinokurov, and L. Mason, "Performance Comparison of OTDM and OBS Scheduling for Agile All-Photonic Network," *IFIP 2005 Conference on Metropolitan Area Networks*, April 2005. Vietnam.

[3] X. Liu, "Time Slotted Scheduling For Agile All-Photonic Networks," Master's thesis, Department of Electrical and Computer Engineering, McGill University, 2005.

[4] T. Anderson, S. S. Owicki, J. B.Saxe, and C. P.Thacker, "High-Speed switch scheduling for Local-Area networks," *ACM Transactions on Computer Systems*, pp. 319–352, Nov 1993.

[5] F. C. Kevin, E. H.-M. Sha, and S. Q. Zheng, "A Fast Noniterative Scheduler for Input-Queued Switches with Unbuffered Crossbars," *8th International Symposium on Parallel Architectures,Algorithms and Networks*, pp. 230–235, 2005.

[6] N. McKeown, "The iSLIP Scheduling Algorithm for Input-Queued Switches," *IEEE/ACM Trans. Networking.*

[7] L. Yihan, S. Panwar, and H. Chao, "On the performance of a dual round-robin switch," *INFOCOM 2001. Twentieth Annual Joint Conference of the IEEE Computer and Communications Societies. Proceedings. IEEE.*

[8] Y. Jiang and M. Hamdi, "A fully desynchronized round-robin matching scheduler for a VOQ packet switch architecture," *Proceedings of the IEEE High Performance Switching and Routing Conference, Dallas, U.S.A*, pp. 407–411, 2001.

[9] D. N. Serpanos and P. I. Antoniadis, "FIRM: A class of distributed scheduling algorithms for high-speed ATM switches with multiple input queues," *Proc. of IEEE Infocom 2000*, pp. 548–555, 2000.

[10] F. J. Gonzlez-Castao, R. Asorey-Cacheda, C. Lpez-Bravo, P. S. Rodrguez-Hernndez, and J. M. Pousada-Carballo, "On the behavior of PHM distributed schedulers for input buffered packet switches," *IEEE Trans. Commun.*, vol. 51, pp. 1057– 1060, July 2003.

[11] F. J. Gonzlez-Castao, R. Asorey-Cacheda, C. Lpez-Bravo, P. S. Rodrguez-Hernndez, and J. M. Pousada-Carballo, "Analysis of Parallel Hierarchical Matching Schedulers for

Input-Queued Switches under Different Traffic Conditions," *Eighth IEEE Symposium on Computers and Communications*, p. 527, 2003.

[12] D. Serpanos, P. Mountrouidou, and M. Gamvrili, "Evaluation of Hardware and Software Schedulers for Embedded Switches," *ACM Transactions on Embedded Computing Systems (TECS)*, vol. 3, pp. 736 – 759, 2004.

[13] P. Gupta and N. McKeown, "Design and Implementation of a Fast Crossbar Scheduler," *IEEE Micro*, vol. 19, pp. 20–28, Jan. 1999.

[14] http://klamath.stanford.edu/tiny-tera/.

[15] E. Soto, E. Lago, and J. Rodrguez-Andina, "FPGA implementation of high-performance PHM / DPHM schedulers," *Field Programmable Logic and Applications, 2006. FPL '06.*

[16] D. Serpanos, P. Mountrouidou, and M. Gamvrili, "Evaluation of Switch Schedulers for Embedded Systems," *Eighth IEEE Symposium on Computers and Communications*, pp. 541,, 2003.

[17] G. Bochmann, T. Hall, O. Yang, M. Coates, L. Mason, and R. Vickers, "The Agile All Photonic Network: An Architectural Outline," *Queen's Biennial Conference on Communications*, Feb 2004.

[18] R. Vickers and M. Beshai, "PetaWeb Architecture," *Networks 2000 Symposium, Canada*, 2000.

[19] L. Mason, A. Vinokurov, N. Zhao, and D. Plant, "Topological Design and Dimensioning of Agile All Photonic Networks," *Computer Networks, Special issue on Optical Networking.*

[20] M. Karol, M. Hluchyj, and S. Morgan, "Input versus Output Queueing on a Space Division Switch," *IEEE Trans. Commun.*

[21] Y. Tamir and G. Frazier, "High-performance multi-queue buffers for VLSI communications switches," *ACM Special Interest Group on Computer Architecture*, pp. 343 – 354, 1988.

[22] Douglas Brent, 'Introduction to Graph Theory', 2nd edition, Prentice Hall,1996.

[23] Y. Tamir and H. Chi, "Symmetric crossbar arbiters for VLSI communication switches," *Parallel and Distributed Systems, IEEE Transactions.*

[24] R. O. LaMaire and D. N. Serpanos, "Two-dimensional round-robin schedulers for packet switches with multiple input queues," *IEEE/ACM Trans. Networking.*

[25] D. Pan and Y. Yang, "Hardware efficient two step iterative matching algorithms for VOQ switches," *Parallel and Distributed Systems, 2006. ICPADS 2006.*

[26] N. McKeown and T. E. Anderson, "A quantitative comparison of iterative scheduling algorithms for input-queued switches," *Computer Networks and ISDN Systems.*

[27] OPNET, http://www.opnet.com.

[28] W. E. Leland, W. Willinger, D. V. Wilson, and M. S. Taqqu, "On the Self-Similar Nature of Ethernet Traffic(extended version)," *IEEE/ACM Trans. Networking.*

[29] J. Potemans, B. V. den Broeck, Y. Guan, J. Theunis, E. V. Lil, and A. V. de Capelle, "Implementation of an advanced traffic model in OPNET modeler," *OpnetWork 2003, Washington D.C., USA*, August 2003.

[30] P. Leys, J. Potemans, B. V. den Broeck, J. Theunis, E. V. Lil, and A. V. de Capelle, "Use of the Raw Packet Generator in OPNET," *OpnetWork 2002, Washington D.C., USA*, 2002.

[31] http://www.altera.com.