Dynamic Distribution Of Memory For Switch Architecture

Ms. Vishnu Priya. A[1], Dr. Senthil Kumar P[2,]

Research Scholar[1], Professor[2],
Anna university Chennai[1], SKR Engineering College Chennai[2], India
[1]vishnu_vishnupriya@yahoo.co.in

Abstract: Interconnection networks are a key component of a variety of systems. In real time, low-latency and contention-free interconnection networks are demanded for the execution of many applications in systems . In modern interconnection networks it is mandatory the use of an effective congestion management technique in order to keep network performance at maximum level under any situations. Although congestion may be avoided by scaling the network size, but the current trends are to reduce overall equipment cost and power consumption of a network ,by plummeting the number of network components. Thus, the network will be prone to congestion, thereby becoming congestion free is mandatory for an efficient & effective network . Therefore, in this dissertation we describe the new congestion management technique (RECN-IQDD) suitable for any type of IQ (Input queue switch architecture: only queues at input port of a switch) switches with enhanced RECN(Regional Explicit Congestion Notification: an efficient Head-of-Line block elimination technique ,with a cost effective Switching architecture to face the challenges of congestion management, has been recently proposed for Advanced Switching (AS) .The idea behind RECN-IQDD is, starting with a simple input queued switch with a single queue per input port, to add some extra queues dynamically allocated for storing congested packets ,to avoid HOL blocking and Distributed deallocates of set aside when congestion vanishes . so, HOL blocking is completely eliminated with less number of queues . Regarding the performance it leads to a significant reduction of the data memory area required at each port in the reduction factor of 5 times than RECN-CIOQ (Combined Input Output Queue - have queues at both Input port & Output port of a switch) and avoids the use of explicit congestion notifications and token-exchanging packets.
[Vishnu Priya. A, Senthil Kumar P. **Dynamic Distribution Of Memory For Switch Architecture.** *Researcher* 2013; 5(8):20-29]. (ISSN: 1553-9865). http://www.sciencepub.net/researcher. 6

Key words: Advanced switching; Congestion,; Buffering; Switching; Flow control

1. Introduction

Usage of Internet is continuously growing at a high rate . Every one is introducing a new businesses on the Internet, new services that are offered through online transactions , and new ways that the Internet can be utilized. Regarding the need and efficiency in interconnection network, problem arrived are the increasing power consumption and cost utilization . For this problem the only solution is to reduce the no of components or resources utilization, because of this reduction link utilization between the every components are increased[4]. So it will automatically guide a need of congestion management. In now a day's Congestion Management in interconnection networks is a widely known problem. Related to this the some of Congestion eliminating techniques are researched recently. Data passes through communication Networks, in the form of packets. These packets contains the information of a source and a destination, and travel in the network through intermediate nodes. When different packets request the same resource, typically in output congestion will occurs. The network grants access to only one packet to a output, while the rest wait.[6] When contention persists, the buffers containing the blocked packets fill, and in lossless networks, which do not allow dropped packets, flow control prevents other switches from sending packets to the congested ports. Although flow control is essential to avoid dropping packets, it rapidly propagates congestion to other switches, because it will also block packets stored at some of their ports. Through these aggregation effects, congestion can spread progressively through the network, forming congestion trees, whose branches match the paths followed by data flows contributing to the congestion (hot flows).[4]Packets belonging to hot flows prevent other packets from advancing, even those belonging to non-congested flows (cold flows). This phenomenon, known as head-of line (HOL) blocking, occurs when a packet at the head of a first - in, first - Out (FIFO) queue becomes blocked, prevents the remaining packets in the same queue from advancing. The impact of HOL blocking on network performance can be very serious: Network throughput can degrade and packet latency can increase dramatically. Network throughput not

only diminishes severely (by approximately 70 percent) [5]when congestion appears, it also remains low well beyond the end of the hot-spot traffic generation. This degradation is the result of HOL blocking produced by the congestion tree affecting other (non-congested) packets. Latency increases from a few hundred nanoseconds (before congestion) to several hundred microseconds, an increase of three orders of magnitude. Packet latencies eventually return to the low values, but only after several milliseconds. This data also confirms that packets experience massive HOL blocking during and after the hot-spot injection. Once this kind of blocking has occurred, even under relatively benign traffic patterns, congestion trees can remain in the network and keeps it operating at reduced efficiency for extended periods of time.[4] Taking all this into account, most designers today choose to reduce system cost and power consumption by reducing the number of network components. If we are to maintain a system's computational power, there are only two ways to reduce the number of network components. If we are to maintain a system's computational power, there are only two ways to reduce the number of network components: increasing the number of end nodes attached to each switch, or using a more suitable fabric topology. However, each of these solutions leads to a higher link use level, thereby driving the network closer to its saturation point and increasing the likelihood of congestion but only the Regional Explicit Congestion Notification (RECN) mechanism achieves both the required efficiency and the scalability that emerging systems demand.[9]

2. Related work

The congestion relevant remedial action in traditional network is just drop the congested packet, when congestion vanish or resources become free it will retransmit the same .this scheme might be lead to data lose ,so The problems associated with congestion are more difficult to solve in lossless networks than in lossy networks such as the Internet, because lossless networks cannot drop packets when congestion occurs. However, researchers have proposed many techniques, representing three basic approaches. The first approach, described in the "Classical congestion elimination or reduction" tries to eliminate the congestion either by preventing its, Appearance (proactive techniques)[5] or by acting immediately upon detection of congestion (reactive techniques)and last one is eliminating negative effect of congestion. However, in general, neither of these techniques can effectively eliminate congestion. Proactive congestion management requires knowledge of the network status and application requirements that is not always available but ir provide a Quality of service. The larger the network, the greater the difficulty in knowing network status, and the more conservative the behavior of the mechanism would have to be. The second, reactive technique ,it detect the occurred congestion, and a notification is sent to source sending packets to the network, by receiving the notification ,source reduce the injected traffic inside the network. But its not easy to sent notification, it may delay ,automatically is not a proper solution, But is it scalable. More recent, congestion management approach takes the view that congestion itself is not a problem. Techniques that follow this approach try to eliminate congestion's negative effects mainly HOL blocking rather than eliminating the congestion trees. The idea is that if the technique can completely eliminate HOL blocking, congestion can exist harmlessly. The complete elimination of HOL blocking has been possible, In general, the most effective HOL-blocking elimination techniques use separate queues to store packets belonging to different flows. Some techniques distinguish data flows on the basis of their final destination (HOL blocking elimination at the network level) and use this criterion for mapping packets to queues[3].For an sample, virtual output queuing(VOQ) methodology ,each input port is going to have many queues to store according to output port .Mainly in VOQsw each input port is having queues as its output port,& in VOQnet each input port is having many as end nodes in the network and every incoming packet is stored in the queue assigned to its destination. This most techniques for eliminating network-level HOL blocking are very efficient, but they require heavy use of resources (queues) for networks with many end points. Thus, these techniques don't scale well; their implementation on large networks is very expensive in terms of silicon area. Other techniques try to solve this problem by eliminating HOL blocking at the switch level. In this case, both resource allocation and packet storing in a switch port depend on the switch output port that the packet requests. In this case, both resource allocation and packet storing in a switch port depend on the switch output port that the packet requests. For example, if a network uses VOQs at the switch level (VOQsw), each switch port has as many queues as output ports in the switch, and a packet is stored in the queue assigned to its switch output port. Therefore, the number of queues at each port depends on the number of switch output ports, but not on the number of network endpoints. This confirms that

VOQsw only partially eliminates HOL blocking but this is scalable. On the other hand, VOQnet maximizes network throughput and minimizes packet latency. This situation changed recently, with the proposal of RECN. This technique for eliminating HOL blocking is fully efficient and scalable. In RECN, the first key difference is that, it stores congested flows separately, the previous techniques allocated queues statically, whereas RECN allocates and deallocates queues dynamically. Thus, RECN uses queues only when it necessary. Another difference is that RECN can address congestion at any network point, not only endpoints. This lets RECN accurately identify congested flows and non-congested ones, so it can separate them. Finally, RECN recognizes that packets belonging to non-congested flows can be stored in the same queue without producing significant HOL blocking, even if they follow different routes.

Table 1: Switch Organization comparison

Organization	HOL blocking eliminated	Queues / switch port	Total no of Queues
FIFO	No	1	2S
VOQsw	Partial	S	S*S
VOQnet	Yes	N	N*N
RECN	Yes	1+SAQs	2S+SAQs

Note: S -- No of port per switch
N – No of end nodes in network
SAQ – No of set aside queues

This allows a great reduction in the number of queues that the mechanism requires. Table 1 describes the some characteristics of different switch organization considered ,it also express the calculation of number of queues per switch port .

3. RECN Implementations

However, RECN has been proposed and designed assuming CIOQ switches. Specifically, the RECN version required several detection queues at each port (one per output port in the switch) in order to detect congestion. In detail, whenever a detection queue fills over a given threshold, the output port associated to the detection queue is considered as a congested point. Although this method works accurately, it is expensive as it increases the silicon area required at input ports, even if memory is dynamically managed.

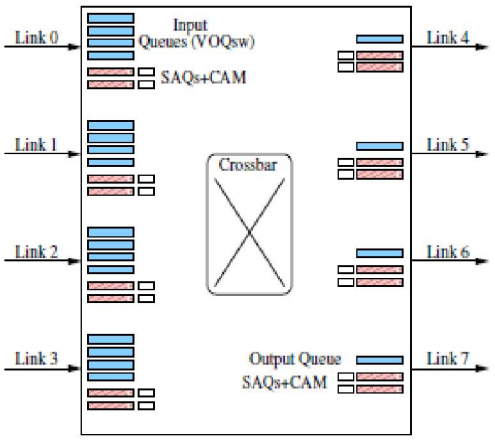

Fig 1: Queue distribution of original RECN –CIOQ

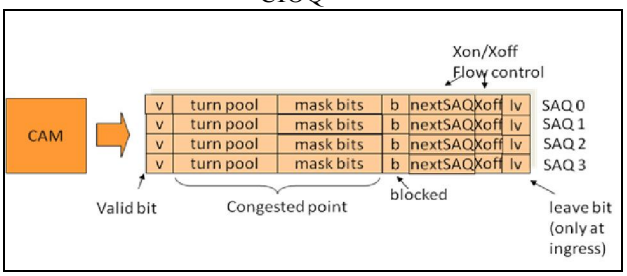

Fig 2: RECN's CAM organization

RECN assumes that packets from non-congested flows can be mixed in the same queue without producing significant HOL blocking. Thus, standard queues will store non-congested packets and SAQs will be dynamically allocated for storing a congested packets . Every set of SAQs is controlled by means of a Content Addressable RECN separates congested and non-congested flows by storing them into different queues, thus eliminating the HOL blocking introduced among them. Specifically, RECN adds a set of additional queues SAQs at every input and output port of a switch. Fig1 shows an schematic of the memory organization required in order to implement RECN on a CIOQ switch architecture.

Memory (CAM) see fig 2 [4], each CAM line containing control information for managing an associated SAQ, including the information required for addressing a congested point. In this sense, RECN addresses, the Source routing to network points by using the routing information included in packet headers. For instance, Advanced Switching (AS) [2] packet headers include a turnpool made up of 31 bits, which contains all the turns (i.e. offset from the input port to the output port) for every switch in a route. Therefore, in AS networks, CAM lines include turnpools addressing

congested points, and these turnpools can be compared to the turnpool of any packet, in order to know whether the packet will cross the corresponding congested point. In this way, packets crossing a particular congested point can be easily detected. In order to identify a point as congested, RECN implements different congestion detection mechanisms at input and output ports. At any output port (see Fig1), whenever the standard queue fills over a given threshold, congestion is detected at this point, and a notification is sent to any input port of the switch sending packets to the congested output port. These notifications include the relative address required to reach the congested point from the input port receiving the notification (a turn in the turnpool). Upon reception of a notification, an input port must allocate a new SAQ and fill the corresponding CAM line with the received turnpool. On the other hand, at input ports, the standard queue is divided into several small detection queues, one per output port (therefore following some kind of VOQsw scheme); this can be seen on Fig 1. Whenever a detection queue fills over a given threshold, congestion is detected at the corresponding output port and a new SAQ associated to this congested point is automatically allocated at the input port. Immediately, all the packets stored in the detection queue are transferred to the SAQ (the detection queue becomes empty and non-operative), and a notification containing the turnpool of the newly allocated SAQ is sent to the output port of the upstream switch, where a new SAQ should be subsequently allocated. Once a SAQ is allocated at a given port, every incoming packet will be directly stored in the SAQ if it will pass through the associated congested point. Otherwise, the packet will be stored in the standard (or detection) queue. In this way, non-congested packets are always separated from the congested ones, thereby preventing the appearance of HOL blocking among them.[3] Furthermore, if any SAQ becomes congested (reaches a given threshold), a notification will be sent upstream, and the receiving input or output port will allocate a new SAQ. This procedure can be repeated until these notifications reach the sources. Therefore, a SAQ will be allocated at every point where otherwise HOL blocking could be introduced. RECN uses a SAQ specific Xon/Xoff (Stop & Go) flow control in order to prevent SAQs from using all the memory space. RECN also detects congestion vanishing at any point, in such a way that SAQs can be deallocated asynchronously. Specifically, the conditions for deallocating a SAQ are the following: the SAQ must be empty, and it must be in Xon state (must not be blocked by a downstream SAQ). Note that these conditions allow a distributed SAQ deallocation, in such a way that a SAQ can be deallocated independently of other SAQs. Since deallocated SAQs can be re-allocated for new congested points, this policy reduces the number of SAQs per port required for completely eliminating HOL blocking. [5]

A. Routing requirements

Like other techniques for eliminating HOL blocking, RECN is also stores the congested packets in separate queues based on congestion tree information . Fig 3, is requirement of switches input port & fig 4 for output port. [5] More precisely RECN detects and Notifies the roots of the different congestion Trees present in the network and later checks at any notified Port whether an incoming packet will pass through one of these roots are separated as different queues. Therefore, RECN should requires the capability to inspect packet routes in advance at all port to achieve efficiency and maximum accuracy , RECN Must also detect congestion inside the network, not only at the endpoint. Therefore, it requires a method of identifying any network port. The use of source routing , RECN meets both requirements. Specifically, RECN assumes the routing properties of Advanced Switching (AS), an open standard for fabric-interconnection technologies developed by the ASI.

B. Congestion notification :

RECN detects congestion in both at input and output ports. At input ports (Fig 3), the standard queue is handled as several detection queues, each one assigned to an specific output port. Non-congested packets are stored in the detection queue associated with their requested output port2. Whenever the occupancy of a detection queue reaches a certain input detection threshold, a new SAQ allocation is triggered at the same input port. This SAQ allocation implies that the valid bit in the corresponding CAM line is set, and the turnpool and bit mask fields are filled in order to address the output port associated with the detection queue. Additionally, the blocked bit (B) is reset. At the output port (Figure 7), if a new packet arrives and the standard queue occupancy is beyond a , given output detection threshold, an input SAQ allocation request is notified to the input port that sent the arriving packet to the output port. This internal notification includes the turnpool and bit mask that identifies the output port from the notified input port. Of course, this information will be stored in the CAM line(Fig 2) associated with the input SAQ that will be allocated upon reception of the notification. Whenever a SAQ at the input port (Fig 3) reaches the Xoff threshold, it raises the sending of an Xoff

flow control packet to the upstream links of the switch. Then the control packet is sent through all the adjacent output port (the output port connected to the link than the input port), and it contains the turnpool and the bit mask associated with the input SAQ. Similarly, once the Xon threshold is reached down in a SAQ, an Xon flow control packet (also containing the corresponding turnpool and bit mask) is sent through the adjacent output port. Upon reception of the Xoff control packet, the output port at the upstream compares the received turnpool and bit mask to all the valid turnpools from the CAM. If there is a matching, then the corresponding SAQ is blocked (B bit is set). If there is no matching, then the Xoff control packet is considered also as a congestion notification and thus, a new SAQ is allocated for the network point indicated by the received turnpool and bit mask. Similarly, whenever a SAQ at the output port (Fig 4) reaches the Xoff or Xon threshold, it broadcasts an internal flow control notification to all the input ports of its switch. This broadcast has the turnpool and bit mask associated with the output SAQ. At each input port, this routing information is updated , include this new turn, is the one required for reaching the output port from the input port. Each input port compares the modified turnpool and bit mask to all the valid turnpools in its CAM. On matching, it handles the internal notification as a flow control notification (the B bit of the corresponding CAM line is set accordingly). If the notification is Xoff type and there is no match, then the input port that sent the last packet to the notifying output port allocates a new SAQ for the network point encoded by the modified turnpool and bit mask. After the notification ,the newly allocated SAQ is blocked by setting its B bit. Note that due to the use of flow control messages for propagating congestion information, RECN does not require specific control messages. Moreover, the Xon/Xoff flow control is included in the Advanced Switching specifications. Therefore, regarding control packets, RECN is fully compatible with AS.

C. Resource Deallocation

A SAQ must be deallocated when its associated congestion tree vanishes. Of course, it is very important to avoid early deallocations, that could introduce HOL blocking. RECN detects that congestion is no longer present in a port when certain conditions are met, and only then the corresponding SAQ will be deallocated. In order to allow a distributed SAQ deallocation, these conditions must be checked from local information. In detail, they are the following:

• The SAQ is empty.

• The SAQ is not blocked due to the ready (R) bit at the CAM line. So, there are no link pointers to this SAQ on any other queue. This condition avoids an empty SAQ being deallocated just after its allocation while the congestion tree is present.

• The SAQ is not blocked due to the Xoff flow control. So, the SAQ must be in the Xon state to be deallocated.

This condition avoids a SAQ to be deallocated while the congestion tree is near (the corresponding downstream SAQ occupancy is over the Xoff threshold). Note that a SAQ does not depend on other SAQs for being deallocated. Therefore, as soon as a SAQ is not necessary (its associated congestion tree has disappeared), it will be deallocated. Of course, immediately after its deallocation, the SAQ can be allocated for other congestion tree. This means that the distributed SAQ deallocation allows a very efficient use of network resources. Note also that the deallocation mechanism does not require any specific control message between SAQs, guaranteeing the mentioned RECN compatibility with Advanced Switching.

4.RECN IQ_DD:

RECNDD-IQ improves the previous RECN version in several ways. Mainly it has buffering queues only in input port , (shown in fig 5) Secondly, SAQs can be deallocated independently of other SAQs allocated for the same congestion tree. Finally, explicit congestion notification packets are no longer needed as congestion notifications are now attached to flow control packets. Both leads to lower requirements of queues and control information.

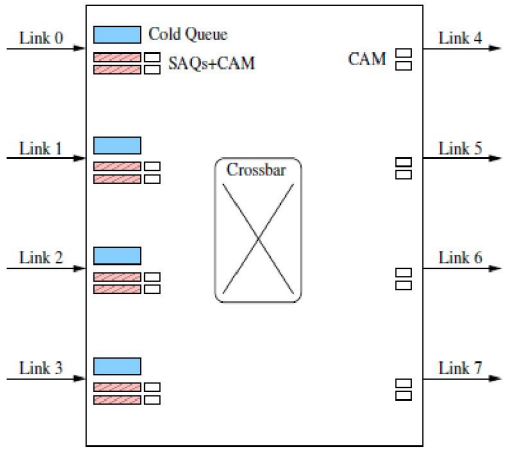

Fig 5: RECN-IQDDs switch architecture

The RECN –IQ mechanism has been designed to have data memory only at input port of a switch. Fig

6 shows the complete architecture of Input Queued switch with RECNDD.RECN_IQ have some different functional units Memory Management Unit (MMU), Congestion Detection Unit(CDU), Routing Unit (RU), Mapping Unit (MU), Packet Processing unit (PPU), Flow Control Unit (FCU), Scheduler And Global flow Control unit.

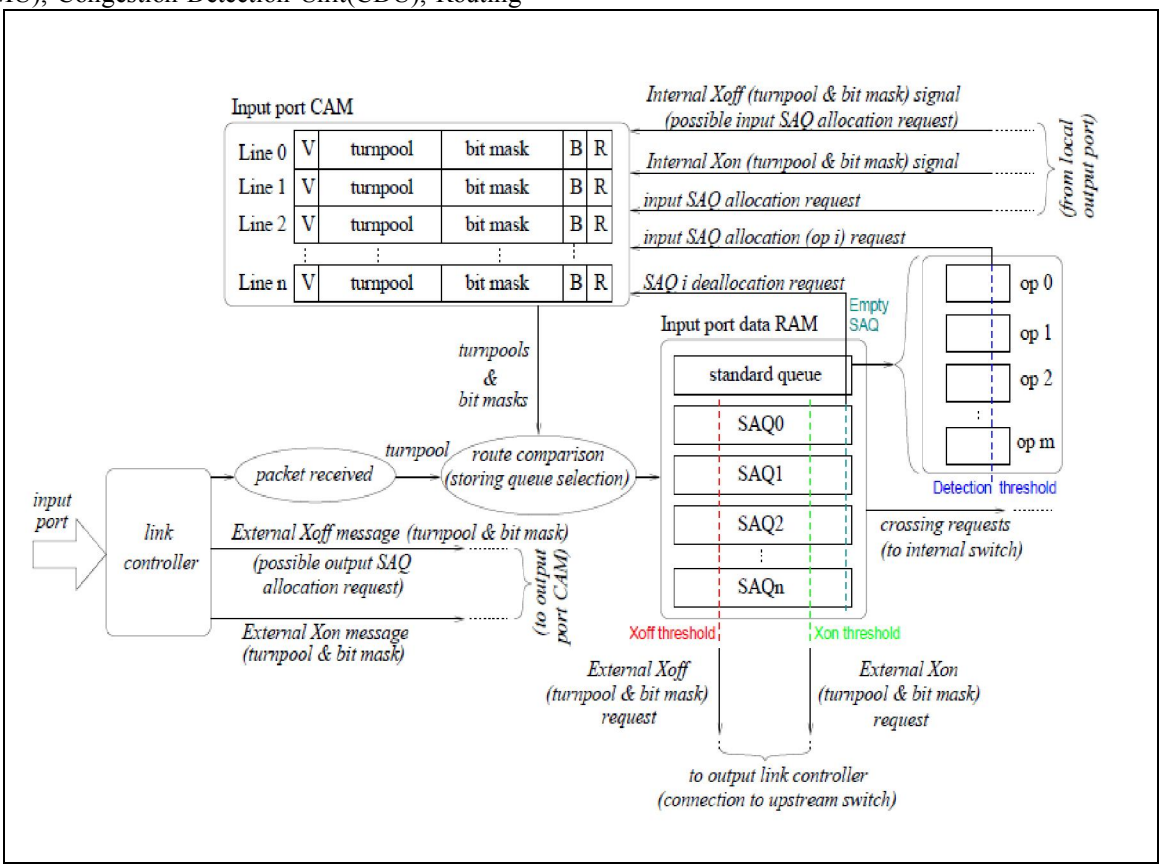

Fig 3: RECN s Input port organization

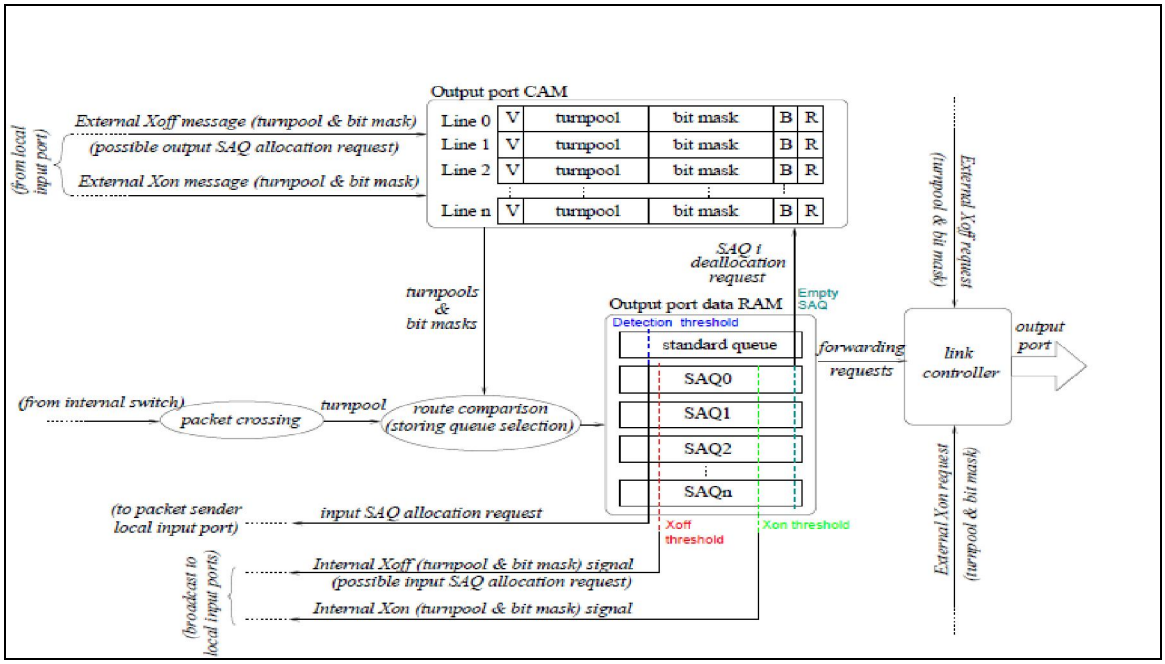

Fig 4: RECNs Output port organization

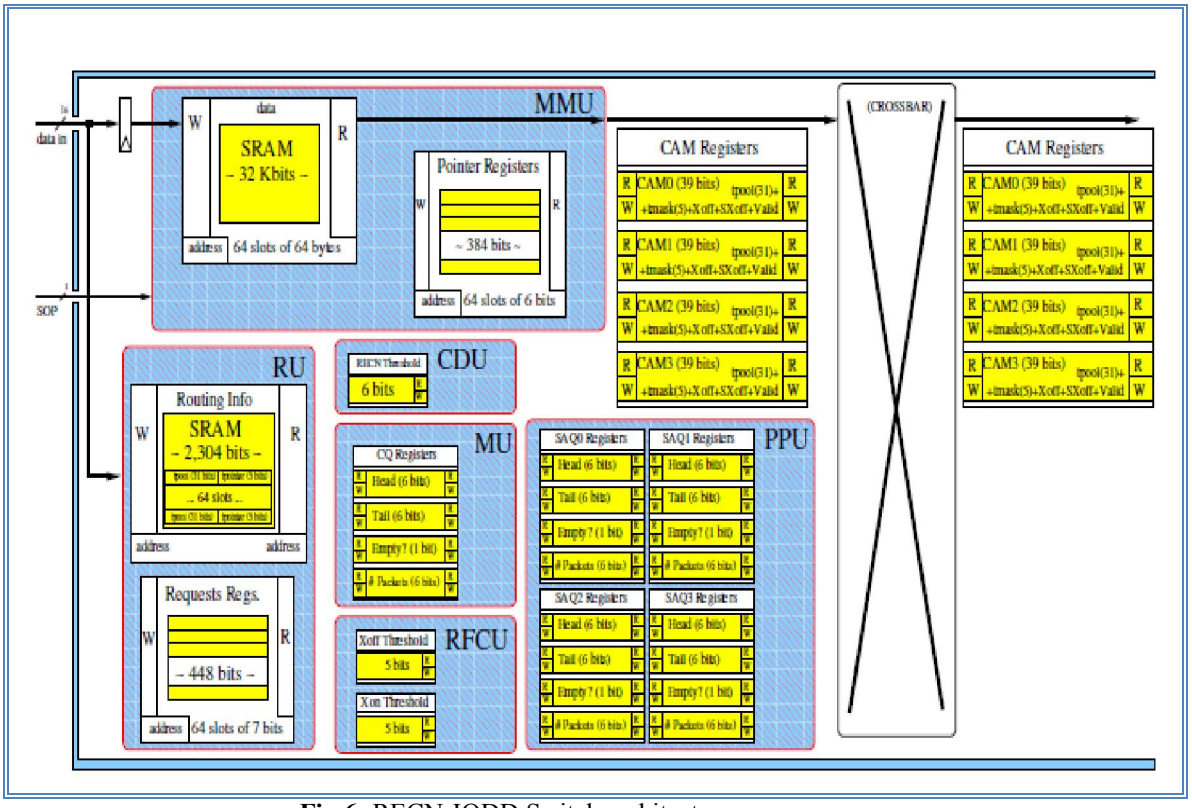

Fig 6: RECN-IQDD Switch architecture

MMU is located inside the Input port.MMU is in charge of mapping all the incoming packets to the corresponding queue and to keep track of the allocated queues within memory It has 64 registers to indicate which the next slot in queues order is. MU is used to keep track of cold queues, which is used to store the newly arrived packet to the input port. RU will decide which output port of switch request according to the header value output port and packet header.CDU is in charge of detecting the congestion, computing the output port congested, and if required allocating the new SAQs for the congested point. PPU have the authority of separating the congested flows from the non-congested one, and it will separate every congested flow from each other. This PPU will move the packet from CQ to a SAQ FCU will supervise Xon/Xoff (Stop & Go) flow control for the SAQs.

This strategy also has the same efficient features like :

- Congestion Detection
- Congestion notification
- SAQ allocation
- SAQ Deallocation
- Packet Processing
- Flow control

A. Congestion detection, notification and SAQ Allocation

With RECN-IQDD, congestion detection is made in the same way as in the original RECN. However, the propagation of congestion detection differs significantly, as congestion notifications are now associated with the Xon/Xoff flow control. Moreover, SAQs for the same congestion tree are not linked (there is one SL field in the CAM, used to store the service level of the corresponding SAQ is assigned). Fig 7 shows CAM structure. For the case of Xon/Xoff flow control between two switches, whenever the occupancy of the input SAQ reaches the Xoff threshold, a Xoff control packet will be sent upstream.

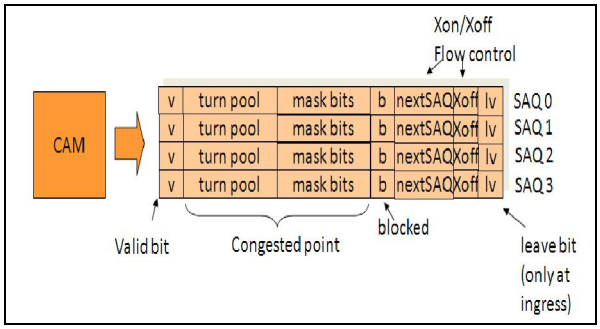

Fig 7: RECN CAM organization

This packet includes the turnpool and the bit mask associated to the SAQ. The upstream output port will compare the received turnpool and bit mask to those associated to all the allocated SAQs at the output port. When matching, the corresponding SAQ will be set at Xoff state (a Xoff control packet has been received), activating the corresponding Xoff bit. However, when failing to match, the switch will consider the Xoff control packet as a congestion notification. Thus, a new SAQ will be allocated for the received turnpool and bit mask. Furthermore, the new allocated SAQ will be set at Xoff state. For the case of Xon/Xoff flow control within a switch, whenever the occupancy of a SAQ at the output side reaches or is beyond the Xoff threshold, an internal Xoff notification is sent to all the input ports of the switch. This notification contains the turnpool and the bit mask from the associated CAM line.

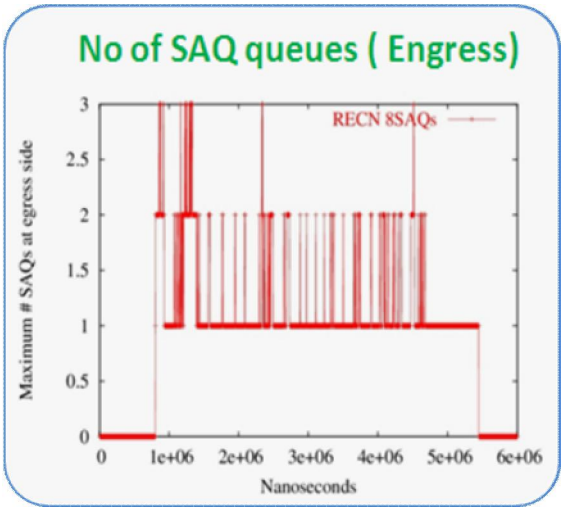

Fig 8: Number of SAQs queues needed at egress of RECN switch mechanism

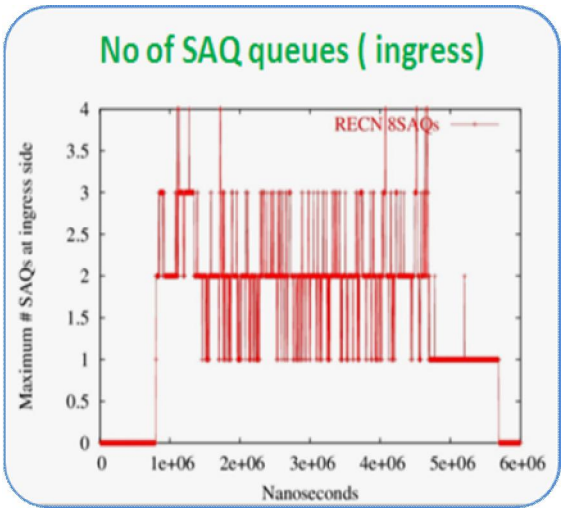

Fig 9: Number of SAQs queues needed at ingress of RECN switch mechanism

The Xoff notification also includes the ID of the input port that sent the last packet to the SAQ. On every input port, upon reception of the internal Xoff notification, the turnpool and bit mask are compared to those associated to all the allocated SAQs. When matching, the corresponding SAQ is set at Xoff state (an internal Xoff notification has been received). If so, the input port must allocate a new SAQ for the congestion tree. Additionally, the newly allocated SAQ will be set at Xoff state. Note that Xon control packets (and Xon internal notifications) also contain a turnpool and a bit mask. This is needed in order to identify which SAQ must be set at Xon state.

B. SAQ deallocation mechanism

With RECN-DQ, SAQs can be deallocated in a distributed way. Thus, the basic deallocation requirement is that the SAQ is empty. However, a new safety condition must be met in order to deallocate SAQs properly: the SAQ is neither blocked due to the blocking bit nor at Xoff state. Note that, since a SAQ is at Xoff state at the moment it is allocated, this condition avoids an empty SAQ being deallocated just after its allocation (premature deallocation). Thus, the timer is no longer needed. Moreover, as the SAQ must be at Xon state to be deallocated, SAQs are not deallocated while congestion is near (a SAQ at Xoff state means that the downstream SAQ is full of packets). Note that all of these conditions can be evaluated completely from local information, and all of them are independent of the state of other SAQs, with the only exception of flow control. Notice also that the external data required for evaluating the new conditions are just those contained in the Xon/Xoff flow control packets. Thus, RECN-DQ does not require specific RECN messages (detection notifications, tokens, etc.). As the Xon/Xoff flow control is considered in the AS specification, RECN-DQ does not require any special control message that is not compatible with AS.

V. Simulation results

In this section, the RECN-DQ mechanism is evaluated by means of simulation results. Specifically, we show the network throughput achieved when the injected traffic is varied. Also, results of switch efficiency as a function of time are presented. These metrics have been measured for different values of the maximum number of SAQs available at input ports. Specifically, we have considered 2, 4 and 8 SAQs for simulating different RECN-DQ configurations and also no SAQs for simulating switches not using RECN-DQ. Regarding network size and topology, the following two Bidirectional Multi-stage Interconnection Network (BMIN) configurations have been analyzed:

• Configuration 1: 64 end nodes, BMIN made of 8x8 switches (48 switches in 3 stages, perfect shuffle as interconnection pattern).

• Configuration 2: 256 end nodes, BMIN made of 8x8 switches (256 switches in 4 stages, perfect shuffle as interconnection pattern).

Other common parameters used in all the simulations are:

- ✓ Input Memories Size=4KB (64 packets);
- ✓ Packet Length=64 bytes;
- ✓ Xon Threshold=5 packets;
- ✓ Xoff Threshold=10 packets;
- ✓ Congestion Detection Threshold=5 packets

A. Results for Uniform Traffic:

For the two BMIN configurations considered, throughput and latency results obtained using uniform traffic and different numbers of SAQs per port can be seen in Fig 8 in egress point, Fig 9 as in ingress point & fig 10 as number of active SAQs. In the case of the 64-end node MIN, the maximum efficiency for an IQ switch without RECN-DD (no SAQs used) is about 65%. When using RECN-IQ and only 2 SAQs, the efficiency increases to above 80%. However, as can be seen in the figures, at least 4 SAQs are needed for achieving maximum switch efficiency

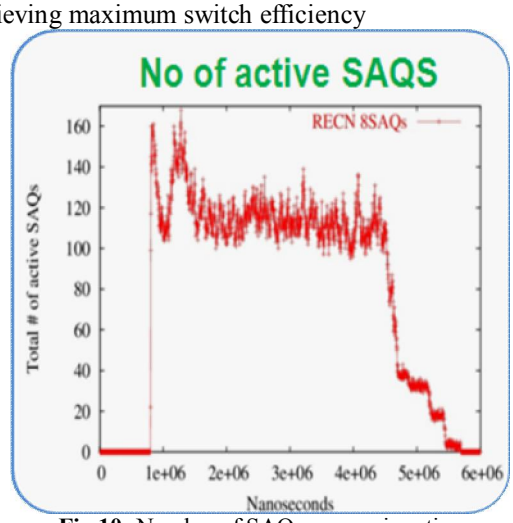

Fig 10: Number of SAQs queues is active

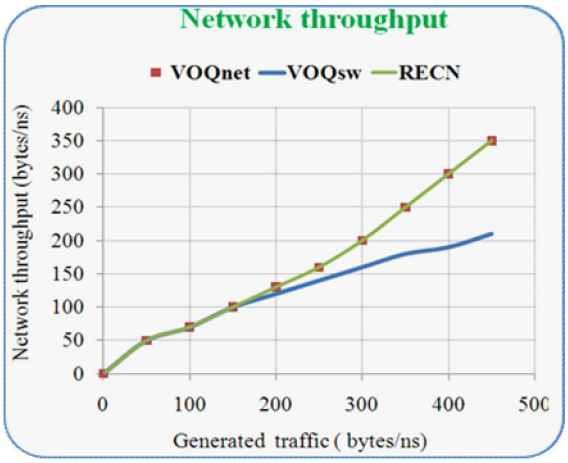

Fig 11: Network throughput of 64*64 networks, with comparison of different switch architecture

The switch efficiency for the 256-end node MIN is slightly lower than the one for the 64-end node MIN, unless we use RECN-IQ and 8 SAQs. In this case, the switch efficiency is maximum. When using only 4 SAQs, the efficiency is above 90%. When using RECN-IQ with only 2 SAQs, switch efficiency can be increased from 65% to almost 80% when compared to the case of not using RECNIQ.

6.Conclusion

This paper concludes a enabled version of RECN , referred to as RECN-IQDD. RECN dynamically separates congested packets from noncongested ones, thus eliminating the HOL blocking. This is achieved by dynamically allocating and deallocating SAQs (Set Aside Queues) for congested packets at switches.

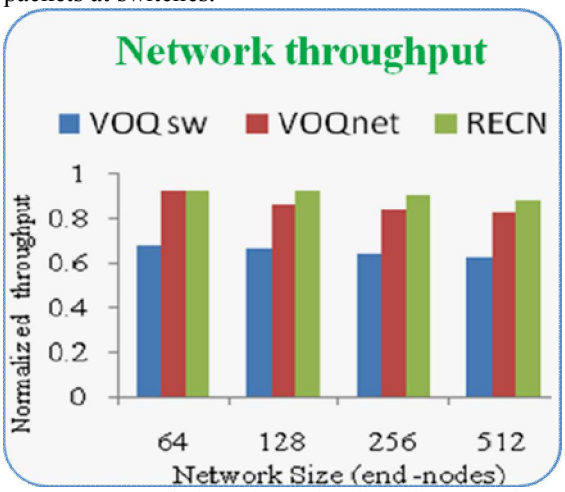

Fig 12: Network throughput of various size networks, with comparison of different switch architecture

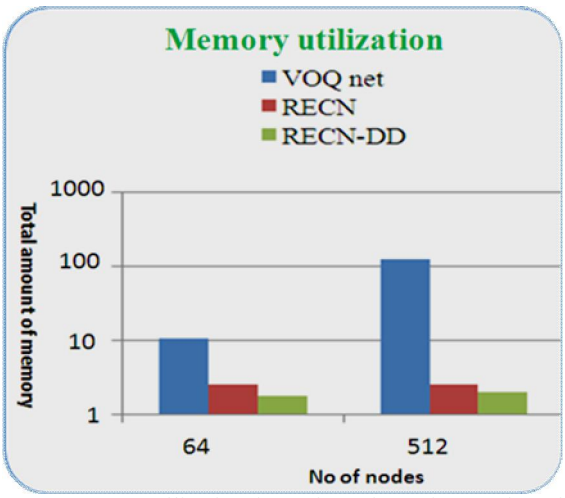

Fig 13: Memory utilization for various sizes of networks, with comparison of different switch architecture

This version introduces a new, completely distributed deallocation SAQ mechanism that does not

impose any constraint on the order in which queues are deallocated. From the performance we can conclude that RECN and RECN-IQ require much less memory area at each port than VOQnet, but they also exhibit an excellent scalability. Note that VOQnet is not scalable at all. Furthermore, RECN-IQDD reduces significantly the memory area required by RECN at each port, thus reducing the overall switch cost. Fig 11,12,13 shows the performance of RECN with compared to others.

REFERENCES:

1. An evolution to crossbar switches with virtual output queuing and buffered cross points by Yoshigoe, K.Christensen, K.J. Univ. of South Florida, FL, USA; Network, IEEE,Sept.-Oct.03 Volume: 7, Issue: 5
2. Hardware implementation of a high-speed Symmetric crossbar switch by Arash Haidari-Khabbaz
3. Achieving 100% Throughput in an Input-Queued Switch by Nick McKeown ,Venkat Anantharam Jean Walrand
4. Input versus Output Queueing on a Space-Division Packet Switch," M.J. Karol, M.G. Hluchyj, and S.P. Morgan, IEEE Trans. Comm., vol. 35, pp. 1347-1356, 1987.
5. "High-Speed Switch Scheduling for Local-Area Networks," ACM Trans. Computer Systems, vol. 11, no. 4, pp. 319-352, Nov. 1993.
6. T. Anderson, S. Owicki, J. Saxe, and C. Thacker, "High-Speed Switch Scheduling for Local-Area Networks", ACM Trans. on Computer Systems, vol. 11, no. 4, pp. 319–352, Nov. 1993.
7. E. Baydal, P. López and J. Duato, "A Congestion Control Mech- anism for Wormhole Networks", in Proc. 9th. Euromicro Work- shop Parallel & Distributed Processing, pp. 19–26, Feb. 2001.
8. L. S. Brakmo and L. L. Peterson, "TCP Vegas: End To End Congestion Avoidance on a Global Internet", IEEE Journal on Selected Areas in Communication, vol.13, no. 8, pp. 1465–1470
9. A Switch Architecture Guaranteeing QoS Provision and HOL Blocking Elimination by Alejandro Martı´nez, Pedro J. Garcı´a, Francisco J. Alfaro, Jose´ L. Sa´nchez,Jose´ Flich, Francisco J. Quiles, and Jose´ Diatom Parallel and distributed systems, vol. 20, no. 1, january 2010
10. L. S. Brakmo and L. L. Peterson, "TCP Vegas: End To End Congestion Avoidance on a Global Internet", IEEE Journal on Selected Areas in Communication, vol.13, no. 8, 1995.
11. W. J. Dally, "Virtual-channel flow control," IEEE Trans. on Parallel and Distributed Systems, vol. 3, no. 2, pp. 194–205, March 1992.
12. J. Dally and H. Aoki, "Deadlock-Free Adaptive Routing in Multicomputer Networks Using Virtual Channels", IEEE Trans. on Parallel and Distributed Systems, vol. 4, no. 4, April 1993.
13. J. Duato, "A New Theory of Deadlock-Free Adaptive Routing in Wormhole Networks", IEEE Trans. on Parallel and Distributed Systems, vol. 4, no. 12, pp. 1320–1331, Dec. 1993.
14. J. Duato, S. Yalamanchili, and L. M. Ni, Interconnection Net- works: An Engineering Approach (Revised printing), Morgan Kaufmann Publishers, 2003.
15. D. Franco, I. Garces, and E. Luque, "A New Method to Make Communication Latency Uniform: Distributed Routing Bal- ancing", in Proc. ACM International Conference on Supercom- puting (ICS99), pp. 210–219, May 1999.
16. P. T. Gaughan and S. Yalamanchili, "Adaptive Routing Pro- tocols for Hypercube Interconnection Networks," IEEE Com- puter, vol. 26, no. 5, pp. 12–23, May 1993.
17. IBM BG/L Team, "An Overview of BlueGene/L Supercom- puter", in Proc. ACM Supercomputing Conference, Nov. 2002.

6/11/2013