International Journal of Innovative Research in Advanced Engineering (IJIRAE) ISSN: 2349-2163
Volume 1 Issue 8 (September 2014) www.ijirae.com

Development of Modified RED AQM Algorithm in Computer Network for Congestion Control

Santosh M Nejakar*
Department of E&TC
Sanjay Ghodawat Institutions, atigre

Sharanabasappa.R R
Department of E&TC
Sanjay Ghodawat Institutions,atigre

Harshavardhan.D.R
Department of R&D
Advaith Research Labs, Shimoga

Abstract— *Congestion control plays an important role in network resource management in very large networks with heavy traffic. Active Queue Management (AQM) algorithms are one of the approaches to resolve the congestion problems. Majority of AQM algorithms mainly focus on single queued links. The input queued switches are limited in throughput and output queued switches require a large speedup factor so our attention is directed towards Combined Input and Output Queued (CIOQ) switches. A simple modification to the RED AQM algorithm is proposed in this paper in order to account for the presence of both input and output queues in the switch. The weighted sum of input and output queue lengths are specifically used as the congestion measure instead of just the output queue length. The average backlog in the switch is significantly reduced in the low speedup region by using our modified algorithm as compared to RED without this modification. Simulations show that the loss rate in the modified RED slightly larger than that in traditional RED but the output queue length in modified RED is tremendously reduced. The congestion measure which is computed using weighting factor results in reduction of the average backlog. Using our modified algorithm simulations indicate improvement in the queue length and switch utilization.*

Keywords— *Active Queue Management (AQM), Combined Input and Output Queued (CIOQ), Dynamic RED (DRED), Stabilized RED (SRED)*

I INTRODUCTION

Consider a network where sources compete for bandwidth and buffer space while being unaware of current state of resources and unaware of each other. In this setting, congestion arises when the demand for bandwidth exceeds the available link capacity. This leads to performance degradation in the network as packet losses increase and link utilization decreases. To avoid these problems, some kind of organization and control of traffic is needed. One way to perform congestion control is to use Active Queue Management (AQM) in the routers. The basic idea behind an AQM algorithm is to sense the congestion level within the network and inform the packet sources about this so that they reduce their sending rate. Many AQM algorithms have been proposed so far, such as RED, GREEN, REM and BLUE. In particular, there have been several modifications to RED. Some references in this area include Dynamic RED (DRED) [9], Adaptive RED, Stabilized RED (SRED), Jacobson et al. and many others. However, these approaches mainly focus on output queued switches. Most practical routers have input queues also, as this would help reduce the required speedup in the switch. The implementation of flow-based AQM algorithms on combined input output queued (CIOQ) switches has been studied in through simulations.

The paper uses the GREEN algorithm; the rates used as input to the algorithm incorporate, in addition to the output queue arrival rates, the rates of the input side VOQs. This idea is important since in a CIOQ switch, congestion affects input queues as well as output queues. In this paper, we look at the impact of input queues on the design of queue-based AQM algorithms. Specifically, we propose a modification of the Random Early Detection (RED) algorithm to account for the presence of the input queue. Instead of using the output queue length as a congestion measure, we use a weighted sum of the input and output queue occupancies. The key advantage of this is that the RED (output) queue need not get filled up as much as in original RED to start reacting to congestion. To study the effect of this change, simulations were performed in network simulator (ns) for a 4 × 4 and a 16 × 16 CIOQ switch with virtual output queues (VOQs). Results show that our fairly simple modification of the RED algorithm significantly reduces the backlog hence reducing the packet queuing delay. Simulations prove that the utilization remains unaffected as compared to the original RED. The weighting factor given to the input queue length in the congestion measure is a critical parameter. We can use it to strike a balance between reducing the average backlog and preventing the loss rate from becoming too high.

OBJECTIVE

- A simple modification to the RED AQM algorithm is proposed in this paper in order to account for the presence of both input and output queues in the switch.
- The weighted sum of input and output queue lengths are specifically used as the congestion measure instead of just the output queue length.

- The average backlog in the switch is significantly reduced in the low speedup region by using our modified algorithm as compared to RED without this modification.

II. LITERATURE SURVEY

We describe a new active queue management scheme, Random Exponential Marking (REM), that aims to achieve both high utilization and negligible loss and delay in a simple and scalable manner. The key idea is to decouple congestion measure from performance measure such as loss, queue length or delay. While congestion measure indicates excess demand for bandwidth and must track the number of users, performance measure should be stabilized around their targets independently of the number of users. We explain the design rationale behind REM and present simulation results of its performance in wireline and wireless networks.

REF : S. Athuraliya, V. H. Li, S. H. Low, and Q. Yin, "REM: Active Queue Management," IEEE Network, vol. 15, no. 3, pp 48-53, May 2001.

Architectures based on a non-blocking fabric, such as a crosspoint switch, are attractive for use in high-speed LAN switches, IP routers, and ATM switches. These fabrics, coupled with memory bandwidth limitations, dictate that queues be placed at the input of the switch. But it is well known that input-queueing can lead to low throughput, and does not allow the control of latency through the switch. This is in contrast to output-queueing, which maximizes throughput, and permits the accurate control of packet latency through scheduling. We ask the question: Can a switch with combined input and output queueing be designed to behave identically to an output-queued switch? In this paper, we prove that if the switch uses virtual output queueing, and has an internal speedup of just four, it is possible for it to behave identically to an output queued switch, regardless of the nature of the arriving tra_c. Our proof is based on a novel scheduling algorithm, known as Most Urgent Cell First. This result makes possible switches that perform as if they were output-queued, yet use memories that run more slowly.

REF : B. Prabhakar and N. McKeown, "On the Speedup Required for Combined Input and Output Queued Switching," Automatica, Vol. 35, pp. 1909-1920, December 1999.

The RED active queue management algorithm allows network operators to simultaneously achieve high throughput and low average delay. However, the resulting average queue length is quite sensitive to the level of congestion and to the RED parameter settings, and is therefore not predictable in advance. Delay being a major component of the quality of service delivered to their customers, network operators would naturally like to have a rough *a priori* estimate of the average delays in their congested routers; to achieve such predictable average delays with RED would require constant tuning of the parameters to adjust to current traffic conditions. Our goal in this paper is to solve this problem with minimal changes to the overall RED algorithm. To do so, we revisit the Adaptive RED proposal of Feng *et al.* from 1997. We make several algorithmic modifications to this proposal, while leaving the basic idea intact, and then evaluate its performance using simulation. We find that this revised version of Adaptive RED, which can be implemented as a simple extension within RED routers, removes the sensitivity to parameters that affect RED's performance and can reliably achieve a specified target average queue length in a wide variety of traffic scenarios. Based on extensive simulations, we believe that Adaptive RED is sufficiently robust for deployment in routers.

REF : Sally Floyd, Ramakrishna Gummadi, and Scott Shenker, "Adaptive RED: An Algorithm for Increasing the Robustness of RED's Active Queue Management,"Technical report, ICSI, August 1, 2001.

This paper describes a mechanism we call "SRED" (Stabilized Random Early Drop). Like RED (Random Early Detection) SRED pre-emptively discards packets with a load-dependent probability when a buffer in a router in the Internet or an Intranet seems congested. SRED has an additional feature that over a wide range of load levels helps it stabilize its buffer occupation at a level independent of the number of active connections. SRED does this by estimating the number of active connections or flows. This estimate is obtained without collecting or analyzing state information on individual flows. The same mechanism can be used to identify flows that may be misbehaving, i.e. are taking more than their fair share of bandwidth. Since the mechanism is statistical in nature, the next step must be to collect state information of the candidates for "misbehaving", and to analyze that information. We show that candidate flows thus identified indeed have a high posterior probability of taking a larger than average amount of bandwidth.

REF : T. Ott, T. Lakshman, L. Wong. "SRED: Stabilized RED,"Infocom,1999.

III. SYSTEM ANALYSIS

EXISTING SYSTEM

AQM refers to a class of algorithms designed to provide improved queuing mechanisms for routers. These schemes are called active because they dynamically signal congestion to sources even before the queue overflows; either explicitly, by marking packets (e.g. Explicit Congestion Notification) or implicitly, by dropping packets. There are two design considerations in any AQM - first, how the congestion in the network is measured and next how this measure is used to compute the probability of dropping packets. Queue based AQMs couple congestion notification rate to queue size. The AQMs currently employed on

the Internet, such as Droptail and RED belong to this category. Another example of a queue based AQM is BLUE. The drawback of this is that a backlog of packets is inherently necessitated by the control mechanism, as congestion is observed only when the queue length is positive. This creates unnecessary delay and jitter. Flow based AQMs, on the other hand, determine congestion and take action based on the packet arrival rate. Some examples are REM and GREEN. For such schemes, the target utilization can be achieved irrespective of backlog.

DISADVANTAGES OF EXISTING SYSTEM

- It was shown to interact badly with TCP's congestion control mechanisms and to lead to poor performance.
- Congested links have high queue size and high loss rate.

PROPOSED SYSTEM

The modification we proposed is to use the sum of input and output lengths as the congestion measure. To be more specific, consider the RED AQM formula for the average backlog. The instantaneous queue length in this formula is replaced by the weighted sum of qoutput and qinput where qoutput is the length of the output queue and qinput is the sum of lengths of all VOQs corresponding to that output. In this way, the input queue length also contributes to the probability of dropping packets. The dropping itself is done only from the output queue, as the AQM is applied only to the output queue.

ADVANTAGES OF PROPOSED SYSTEM

- This modification reduces the average backlog in the switch significantly when speedup is low.

IV. SYSTEM REQUIREMENT

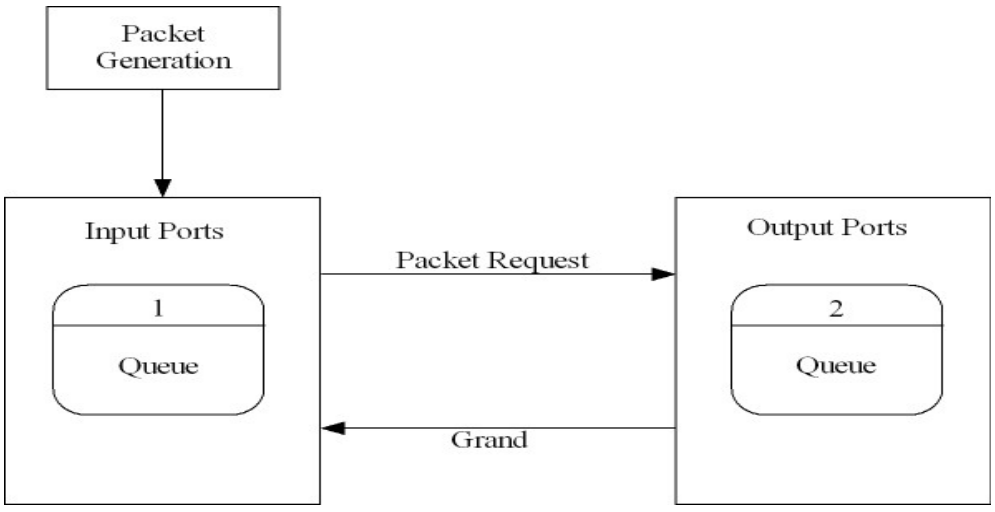
HARDWARE REQUIREMENTS

Processor	:	Intel Pentium III or later
Main Memory	:	256 MB
Cache Memory	:	512 KB
Keyboard	:	108 Keys
Monitor	:	17" Color Monitor
Mouse	:	Optical Mouse
Hard Disk	:	160 GB

SOFTWARE REQUIREMENTS

Front End	:	NS2.
Operating System:	Linux.	

V. SYSTEM ARCHITECTURE



VI. PROBLEM ANALYSIS

QUEUE ARCHITECTURE

In output queuing (OQ) packet switch architectures, packets arriving from input lines are not queued at input interfaces; rather, they cross the switching fabric, and join a queue at the switch output interface before being forwarded to the next node (see Fig. 1.a). In order to avoid contention either within the switching fabric or in the access to the output queue, it is necessary that the internal speed of an OQ switch, as well as the access rate of the memory at each output interface, be equal to the sum of all input line rates (i.e., they must equal the aggregate switch bandwidth). On the contrary, in input queuing (IQ) packet switch architectures, packets arriving from input lines are queued at input interfaces.

They are then extracted from input queues to cross the switching fabric and be forwarded to the next node, according to some algorithm that avoids contention within the switching fabric and at the input and output interfaces (no more than one packet at a time can be extracted from each input and reach each output). Note that the implementation of this algorithm requires coordination among input interfaces, hence the exchange of control information, which can either contend with user data for access to the switching fabric, or use a separate data path. The internal speed of an IQ switch need not be larger than the highest line rate (see Fig. 1.b), possibly incremented to account for extra packet headers added internally to the switch. ¿From this preliminary description of the IQ and OQ approaches, it is immediately evident that the intelligent packet scheduling of IQ compensates for the missing internal speedup of OQ.

However, the comparison of the IQ and OQ approaches deserves deeper investigations. This work was supported by a research contract between CSELT and Politecnico di Torino If FIFO (fist in first out) packet queues are used, IQ architectures suffer from the head of the line (HoL) blocking phenomenon, that severely limits their performance. In 1987, Karol, Hluchyj and Morgan proved with an asymptotic analysis that under source- and destination-uniform traffic, HoL blocking limits the maximum throughput of IQ switches with an arbitrarily large number of input/output lines to 2 □ p2 _ 0:586.

This result is quite disappointing, specially considering that, without any buffering in the switch (except for the obvious need to store one packet at the switch ingress and egress), i.e., by discarding packets that cannot be immediately switched, the throughput limit grows to 0.632. The HoL blocking phenomenon can be circumvented with more complex structures of the input queues, for example separating (either logically or physically) at each input the queues of packets referring to different outputs (see Fig. 1.c), as we shall see.

Moreover, OQ offers the possibility of rather easily controlling the delay of packets within the switch (with algorithms such as fair queuing), since the time needed to cross the switching fabric is known, and all packets following a given route can be found within the same (output) queue. The situation is such that packet switch designs traditionally referred to OQ architectures, especially in the case of ATM (Asynchronous Transfer Mode) switches, because of their greater effectiveness, and because the requirement for an internal speedup was not critical. The recent developments in networking, that produced a dramatic growth of line rates, have however made the internal speed requirements of OQ switches difficult to meet. This has generated great interest in IQ switch architectures, thus opening a lively new research line in switching.

A few implementations of IQ switches exist, either prototypal or commercial. Probably the most famous IQ cell switch prototype is the *Tiny Tera*, implemented at Stanford University in cooperation with Texas Instruments. The latest version of Tiny Tera can reach an aggregate bandwidth of 1 Tbps. A similar architecture is implemented in the recent line of Cisco Gigabit routers, the Cisco 12000 series, which adopts a core IQ switching fabric reaching an aggregate bandwidth equal to 60 Gbps. Other examples of commercial IQ switching fabrics are the Lucent Cajun Switch and the Yago PowerCast Switch.

A number of novel algorithms for IQ switch architectures have recently appeared in the technical literature, aiming at over coming the HoL blocking performance limitations with minimal complexity. In this paper we discuss and compare a number of such proposals in the context of packet switches operating on fixed-size data units. The term *cell* switches will be used, although the discussed switch architectures need not refer to ATM. The considered IQ cell-switch architecture proposals are: iSLIP, iLQF, iOCF, iLPF, 2DRR, RC, MUCS, RPA, and iZIP, a novel proposal.

MULTICAST ROUNG ROBIN SCHEDULING ALGORITHM

With existing three-phase (request-grant-accept) multicast scheduling algorithms that employ the *k*-MC-VOQ architecture, *all* the queues having cells need to issue requests to the destined outputs during the request phase, which means that

a large amount of information needs to be exchanged between inputs and outputs; furthermore, iteration is fundamental to guarantee high throughput performance, which limits the scalability of the scheduling algorithms further as the link speed and port number increase. For this reason, we present a two-phase (request-grant) scheduling with reduced matching overhead where each input selects only one of the *k* HoL cells to participate in the scheduling in each time slot.

Note that since only one HoL cell is chosen for requesting, iteration and accept phase are not required as there are no contentions at each input during the accept phase, however, the performance penalty is expected for such simplification. To overcome this drawback, we propose a novel HoL alleviation scheme in this paper that can still benefit from the *k*-MC-VOQ architecture, and thus reduce the HoL blocking. Input schedulers are distributed at each input, and a global pointer *g* is collectively maintained by all the output schedulers. Each input scheduler maintains two priority pointers: a primary pointer *rp i* and a secondary pointer *rs i* . Primary pointers are designed to provide fairness among *k* multicast queues at each input, while secondary pointers are used to alleviate the HoL blocking, and thus guarantee high throughput performance.

We denote the input as primary input to which the global output pointer points, and the others as secondary inputs. As a result, *N* inputs are divided into two groups: one primary input and $N - 1$ secondary inputs. A detailed description of the algorithm follows, including two steps: *Step 1*: Request. The primary input sends requests to all the destined output ports corresponding to the first nonempty queue in a fixed round-robin order, starting from the current position of its primary pointer *rp i* . The primary pointer of the primary input is incremented to one location beyond the selected queue. Each secondary input sends requests to all the destined output ports corresponding to the first nonempty queue in a fixed round-robin order, starting from the current position of its secondary pointer *rs i* . The secondary pointer of each secondary input is incremented to one location beyond the selected queue. Note that the secondary pointer of primary input and the primary pointer of each secondary input are not updated and remain where they are. *Step 2*: Grant. If an output receives one or more requests, it chooses the one that appears next in a fixed round-robin schedule starting from the current position of the global output pointer *g*. The output notifies each requesting input whether or not its request was granted.

The global output pointer is moved to one position beyond the first input that is served. Fig.2 shows an example of the MDRR arbitration algorithm for a 4×4 switch when $k = 2$. Input 4 is the primary input in the time slot as the global output pointer points to it. All the inputs and outputs perform arbitration in parallel. In a request phase, primary input 4 chooses the 2nd queue to which its primary pointer points and sends requests to output 2 and 3 and updates its primary pointer to 1; with secondary input 2, it chooses the 2nd queue to which its secondary pointer points and sends requests to output 3 and 4, and updates its secondary pointer to 1 whether its requests are granted by destined outputs or not (actually, none of its requests is granted). To consider output 3 in the grant phase, since the global pointer, *g*, is pointing to 4, output scheduler 3 grants input 4. After that, the global pointer is incremented to 1.

VII. PROJECT DESCRIPTION

MODULES

1. Packet Switch Architecture
2. Modified RED Queue
3. Request Phase
4. Grand Phase

1. Packet Switch Architecture

The proposed scheduling algorithm is targeted at synchronous input-queued (IQ) switches. The fixed-size packet transmitted by the switch fabric is also called cell. We consider only the fan-out splitting discipline that cells may be delivered to outputs over several cell times. Any multicast cell is characterized by its fan-out set, i.e., by the set of outputs to which the cell is directed. We define the fan-out size *f* as the number of destinations of a multicast cell. A small number, *k*, of FIFO queues dedicated to multicast traffic are maintained at each input port. Arriving multicast cells are partitioned into the *k* queues according to the fan-out size.

2. Modified RED Queue

The modification we proposed is to use the sum of input and output lengths as the congestion measure. To be more specific, consider the RED AQM formula for the average backlog. The instantaneous queue length in this formula is replaced by the weighted sum of qoutput and qinput where qoutput is the length of the output queue and qinput is the sum of lengths of all

VOQs corresponding to that output. In this way, the input queue length also contributes to the probability of dropping packets. The dropping itself is done only from the output queue, as the AQM is applied only to the output queue.

3. Request Phase

The primary input sends requests to all the destined output ports corresponding to the first nonempty queue. At request phase, fix fanout size of the current non empty queue are measured in each input port and the priorities based on higher size is allotted. Next session is to assign due dates to the cells within the fanout. This Due dates are assigned in a priority input port will assign the first Due Date (Due Date = 1) to the cells obviously. On the second priority port, elements already presented in first priority assigned second Due Date (Due Date = 2) and remaining cells assigned first Due Date (Due Date = 1) and so on. On the completion of these Due Dates, the requests will be made to input ports.

4. Grand Phase

The grand phase receiving the requests and will granted a reject to manage HOL situation. The global pointer increment one in each grand phase. In the case of two requests made for the same port, the global pointer pointing one is granded and other rejected.

VIII. CONCLUSION AND FUTURE ENHANCEMENT

CONCLUSION

CIOQ switches are an important class of switches due to their good throughput and the low speedup required. In this paper, we studied the effect of input queues on the RED AQM. We have proposed the inclusion of the input queue length also in the congestion measure and have shown through simulations that this modification reduces the average backlog in the switch significantly when speedup is low. At the same time, it has been shown that the modified RED has almost the same utilization and loss rate as the original RED algorithm.

FUTURE ENHANCEMENT

Possible extension of this work includes analysis of the performance of the modified algorithm and assessment of its stability. It would also be useful to study analytically how the weighting given to the input and output queue lengths while computing the congestion measure affects the tradeoff between dropping too many packets on one extreme and not including the input queue's effects at all on the other extreme.

REFERENCES

[1] S. Athuraliya, V. H. Li, S. H. Low, and Q. Yin, "REM: Active Queue Management," IEEE Network, vol. 15, no. 3, pp 48-53, May 2001.
[2] M. Karol, M. Hluchyj and S. Morgan, "Input versus Output Queuing an a Space Division Switch," IEEE Trans. Communications, Vol. 35, pp. 1347-1356, 1987.
[3] B. Prabhakar and N. McKeown, "On the Speedup Required for Combined Input and Output Queued Switching," Automatica, Vol. 35, pp. 1909-1920, December 1999.
[4] M. Karol, M. Hluchyj and S. Morgan, "Input versus Output Queuing an a Space Division Switch," IEEE Trans. Communications, Vol. 35, pp. 1347-1356, 1987.
[5] Sally Floyd, Ramakrishna Gummadi, and Scott Shenker, "Adaptive RED: An Algorithm for Increasing the Robustness of RED's Active Queue Management,"Technical report, ICSI, August 1, 2001
[6] J. Aweya, M. Ouellette, D. Y. Montuno and A. Cha-- pman, "An Optimization-oriented View of Random Early Detection," Computer Communications, 2001
[7] V. Jacobson, K. Nichols, and K. Poduri, "RED in a Different Light," Technical Report, September 1999
[8] T. Ott, T. Lakshman, L. Wong. "SRED: Stabilized RED,"Infocom,1999

