Available online at www.sciencedirect.com
SCIENCE DIRECT
Computer Communications 28 (2005) 300–312
computer communications
www.elsevier.com/locate/comcom

A simulation study of the Adaptive RIO (A-RIO) queue management algorithm

Julio Orozco[a,b], David Ros[a,*], José Incera[c], Rodolfo Cartas[c]

[a]GET/ENST Bretagne, rue de la Châtaigneraie, CS 17607, 35576 Cesson Sévigné Cedex, France
[b]IRISA/INRIA Rennes, Campus de Beaulieu, 35042 Rennes Cedex, France
[c]ITAM, Río Hondo No. 1, Sn. Angel, 01000, México D.F., Mexico

Received 14 October 2004; accepted 14 October 2004
Available online 28 November 2004

Abstract

Active queue management (AQM) algorithms are useful not only for congestion avoidance purposes, but also for the differentiated forwarding of packets, as is done in the DiffServ architecture. It is well known that correctly setting the parameters of an AQM algorithm may prove difficult and error-prone. Besides, many studies have shown that the performance of AQM mechanisms is very sensitive to network conditions. In this paper we present a detailed simulation study of an Adaptive RIO (A-RIO) AQM algorithm which addresses both of these problems. A-RIO, first introduced by Orozco and Ros (2003), draws directly from the original RIO proposal of Clark and Fang (1998) and the Adaptive RED (A-RED) algorithm described by Floyd et al. (2001). Our results, based on ns-2 simulations, illustrate how A-RIO improves over RIO in terms of stabilizing the queue occupation (and, hence, queuing delay), while maintaining a high throughput and a good protection of high-priority packets; A-RIO could then be used for building controlled-delay, AF-based services. These results also provide some engineering rules that may be applied to improve the behaviour of the classical, non-adaptive RIO.

Keywords: Active queue management; DiffServ; IP networks; Quality of service

1. Introduction

Active queue management (AQM) is the name given to a type of router mechanisms used in congestion control. AQM mechanisms manage queue lengths by dropping[1] packets when congestion is building up [1], that is, before the queue is full. End-systems can then react to such losses by reducing their packet rate, hence avoiding severe congestion. Random Early Detection (RED) [2] is one of the first AQM mechanisms to have been proposed, and the one that has been most studied. RED intends to avoid congestion by randomly discarding packets based on the average queue size.

AQM mechanisms are also relevant in the context of DiffServ. The DiffServ architecture has been defined by the IETF to provide IP networks with scalable QoS processing of traffic aggregates, based on a special field in the IP header [3]. The value of this field tells a router what particular treatment, the per-hop behaviour (PHB), should be applied to the packet.

One of the standard PHBs is Assured Forwarding (AF) [4]. AF defines the differentiated forwarding of packets classified in up to four classes. Packets from different classes are processed in different physical queues, managed by a scheduling mechanism. Within each class (or queue), there can be up to three drop precedences. Under congestion, packets marked with the highest priority—that is, the lowest drop precedence—should be the last to be discarded, and vice versa. Such differentiated discarding inside a single queue can be achieved by means of special AQM mechanisms, which are usually extensions of RED.

* Corresponding author. Tel.: +33 2 99 12 70 46; fax: +33 2 99 12 70 30.
E-mail addresses: julio.orozco@irisa.fr (J. Orozco), david.ros@enst-bretagne.fr (D. Ros), jincera@itam.mx (J. Incera), rodolfo@mcculloch.cannes.itam.mx (R. Cartas).

[1] Or marking, if a mechanism such as Explicit Congestion Notification (ECN) is used.

0140-3664/$ - see front matter © 2004 Elsevier B.V. All rights reserved.
doi:10.1016/j.comcom.2004.10.013

In this paper we describe an adaptive AQM algorithm, which we call Adaptive RIO (A-RIO), suitable for building an AF per-hop behaviour. A-RIO is a straightforward combination of the Adaptive RED (A-RED) algorithm described by Floyd et al. [5] and the RIO algorithm of Clark and Fang [6]. The goal of A-RIO is threefold: (1) to simplify the configuration of DiffServ-enabled routers, by alleviating the parameter setting problem most AQM algorithms present; (2) to automatically translate a quality-of-service parameter (that is, delay) into a set of router parameters; (3) to try to stabilize queue occupation around a target value under heavy network load, irrespective of traffic profile.

This paper is organized as follows. In Section 2, we discuss the AQM and DiffServ issues that define the context, motivation and basis of our proposal. In Section 3, the A-RIO algorithm is described in detail. In Section 4, we report on the process and results of a simulation study of A-RIO. Conclusions and perspectives are provided in Section 5.

2. Active queue management in DiffServ networks

The main goal of Active Queue Management (AQM) mechanisms in best-effort networks is congestion avoidance. The well-known RED algorithm avoids congestion by controlling the average queue size and comparing it against two thresholds, min_{th} and max_{th}. Within this region, packets are discarded using a linear probability function of the average queue size. Besides min_{th} and max_{th}, RED requires to set up two other parameters: a maximum drop probability max_p for early discard and an absorption factor used in the average queue size function w_q. To the best of our knowledge, there are no precise rules for tuning these parameters; on the contrary, most published results point at the difficulty of finding a robust RED configuration (see for instance [7,8]).

In a DiffServ environment, the AQM mechanisms are used mainly for prioritized discard. RED with In and Out (RIO) [6] is the basic mechanism to set up an AF PHB. RIO is a direct extension of RED that uses two sets of parameters to differentiate the discard of *In* (in-profile) and *Out* (out-of-profile) packets. For deciding whether to discard *Out* packets, RIO uses the average size of the total queue, formed by *In* and *Out* packets. For *In* packets, it uses the average size of a virtual queue formed only by *In* packets. RIO has been extended to handle $n>2$ precedences following the same principles.[2] The discard probability for packets of precedence $1 \le j < n$ depends on the average size of a virtual queue containing only the packets with precedences 1 to j. For packets with precedence n (i.e. the lowest priority), the discard probability is a function of the average occupation of the 'physical' (total) queue. This method was eventually called RIO-C (*Coupled*) to differentiate it from others proposed later. For example, Weighted RED (WRED) [9] uses the total average queue size for all precedences, whereas RIO-DC (*Decoupled*) [10] calculates the drop probability for packets of precedence j as a function of the average number of packets of the same precedence only.

RIO-C discriminates packets of different precedences in three ways. The first one lies in the coupled calculation of the discard probability; the fact that the discard probability for precedence j uses the average number of packets of *all* lower precedences yields a significant discrimination. The second way is the use of different thresholds for different precedences, so that discard begins 'earlier' for packets of higher precedences. The third one is the use of drop probabilities that increase at different rates for different priorities. Note that the last two ways of achieving differentiation are based simply on different parameter settings, and that they are not mutually exclusive.

The parameter setting problem gets magnified with RIO: for a n-precedence RIO, in principle $3n+1$ parameters should be set: $2n$ thresholds, n maximum drop probabilities, and the averaging weight w_q—assuming it is the same for all virtual queues. This problem has become a subject of research. Studies such as [11] illustrate the difficulty of tuning RIO in order to achieve a predictable performance. This issue is very relevant, since a key idea behind DiffServ is to allow the provisioning of enhanced services. The operator should know how to set up services with some (loose) guarantees in rate or delay.

3. An adaptive RIO (A-RIO) algorithm

Different approaches have been proposed in the literature for dealing with the tuning of RED (see for instance [8,12]). The one taken by the Adaptive RED (A-RED) [5] algorithm is to dynamically adjust RED's operation parameters based on the measured traffic load at the router.

A-RIO, first introduced in [13], is a direct extension of both A-RED and RIO-C algorithms. It follows the approach of the former, performing an on-line automatic adaptation of the mechanism to get a more predictable performance, together with a more straightforward parameter tuning. We have chosen A-RED because of its simplicity, both in concept and in implementation.

With this proposal, we intend to alleviate the problem of parameter tuning in the context of DiffServ AF networks. Like A-RED, A-RIO needs a single input parameter, the *target delay*, which is translated into the required set of router parameters. This feature could be very interesting for providers of differentiated services: configuring routers in terms of delay—a QoS metric directly related to service specifications and customer requirements—should be much

[2] In the $n=3$ case, packet drop precedences are usually identified by colors: *green* for the lowest precedence, *yellow* for the middle one and *red* for the highest one.