

國立交通大學

資訊工程學系

博士論文

應用於終端設備以提供 TCP 相等性
與閘道器上以排程請求回應
的公平控制方法

Fairness Controls for
TCP-Equivalence at Endpoint and
Request-Response Scheduling at Gateway

研究生：曹世強

指導教授：林盈達 教授

中華民國九十六年十二月

應用於終端設備以提供 TCP 相等性
與閘道器上以排程請求回應
的公平控制方法

Fairness Controls for
TCP-Equivalence at Endpoint and
Request-Response Scheduling at Gateway

研究生：曹世強
指導教授：林盈達

Student : Shih-Chiang Tsao
Advisor : Prof. Ying-Dar Lin



A Dissertation Submitted to
Department of Computer Science
College of Computer Science
National Chiao Tung University
for the Degree of
Doctor of Philosophy
in
Computer Science
December 2007
Hsinchu, Taiwan, Republic of China

中華民國九十六年十二月

應用於終端設備以提供 TCP 相等性 與閘道器上以排程請求回應 的公平控制方法

學生：曹世強

指導教授：林盈達

國立交通大學資訊工程學系博士班

摘 要

為了在網路上傳送封包，資料流可能需要為頻寬而競爭。當資料流為 Internet 上頻寬競爭時，公眾公平性是需要被維持的，相對的，當資料流於私有的接取路徑上競爭時，則私有公平性可能需要被維持。為維持公眾公平性，不同於 TCP 的速度控制方法必須使用不超過 TCP 資料流的頻寬。然而，這些方法常只使用低於 TCP 資料流的頻寬來保守的達到公眾公平性。在另一方面，為維持私有公平性，使用封包排程器來管理瓶頸路徑是一個常見的方法。但是，這方法無法在使用者端的接取閘道器上管理呈現瓶頸狀態的下載路徑，因為他不能控制或排程那些在使用者端對面之 ISP 端閘道器等候的封包。

我們首先針對八個知名的速度控制方法，探討其為何無法恰巧使用與 TCP 相等的頻寬，也就是表現出 TCP-equivalence 的特性。接著我們提出了一個 window-averaging rate control (WARC)方法。藉由只考慮固定區間內的 TCP 速度，使得 WARC 能夠更早的拋棄歷史封包遺失狀態，因而能表現出比過去方法較好的 TCP-equivalence 特性。最後，我們又提出了 minimum-service first request scheduling (MSF-RS)的方法來解決封包排程器無法在使用者端管理下載路徑的私有公平性問題。MSF-RS 藉由排程上行路徑請求以控制下行路徑回應的方式，來達到使用者為基礎的權重公平性，也就是無論一類別使用者數量多寡，都能確保高等級類別使用者獲得較多的頻寬。

模擬結果分別在非週期性封包遺失，低耦合度流量，雙狀態遺失，及突現性

大量遺失四種狀態下，顯示出先前速度控制方法無法在 TCP-equivalence 情況下維持公眾公平性的原因，而分析及模擬結果也顯示 WARC 能藉由更快的加減速反應，來表現更好的 TCP-equivalence 及達到公眾公平性。最後分析模擬及實驗結果顯示 MSF-RS 能在使用者閘道器上提供以使用者為基礎的私有權重公平性，並縮短 20~30%的使用者感受延遲時間。

關鍵字- TCP 友善性, 壅塞控制, 請求排程, 接取閘道器, 公平佇列



Fairness Controls for TCP-Equivalence at Endpoint and Request-Response Scheduling at Gateway

Student : Shih-Chiang Tsao

Advisors : Dr. Ying-Dar Lin

Department of Computer Science
National Chiao Tung University

Abstract

Flows may compete for bandwidth to transmit packets. *Public fairness* should be maintained by the flows when they compete for the bandwidth in the Internet, while *private fairness* may be required when they do at a private access link which connects the intranet to the Internet. To maintain the public fairness, rate control schemes different from TCP should use no more bandwidth than TCP. However, these schemes often only use less bandwidth to conservatively maintain the fairness. On the other hand, for maintaining the private fairness, the usual solution is using a packet scheduler to manage bottleneck. Nevertheless, the solution fails to manage the downlink bottleneck at the user-side access gateway, since it cannot schedule the packets queued at the ISP-side gateway, opposite to the user-side one.

This dissertation first investigates eight well-known rate control schemes to reveal why they cannot maintain the public fairness by using just the same bandwidth as TCP, i.e. being TCP-equivalent. Next, this dissertation proposes a window-averaging rate control (WARC) scheme. Considering the TCP rate only over a fixed interval leads WARC to forget the historical packet loss condition more quickly and thus perform better TCP-equivalence than other schemes. Finally, a minimum-service first request scheduling (MSF-RS) scheme is proposed to solve the private fairness problem which packet schedulers fail to manage downlink at the user-side gateway. MSF-RS schedules uplink requests to control downlink responses in order to provide user-based weighted fairness, i.e. ensure high-class users to get more bandwidth even more users belong to the high class.

The simulation results under non-periodic losses, low-multiplexing, two-state losses, and bursty-losses reveal the causes that previous schemes cannot maintain

public fairness with TCP-equivalence. Next, both analysis and simulation demonstrate that WARC does maintain the fairness and perform better TCP-equivalence by exhibiting the faster aggressive and responsive behaviors. Finally, the analysis, simulation and field trial exhibit that MSF-RS provides the user-based private weighted fairness while reducing 20~30% of user-perceived latency at the user-side gateway.

Keywords- TCP-friendly, congestion control, request scheduling, access gateway, fair queuing



Table of Content

1	Introduction	1
1.1	Bottlenecks for the Internet Traffic	1
1.2	Public Fairness Control for TCP-Friendliness	1
1.3	Private Fairness Control for Weighted Fairness	2
1.4	Related Work and Potential Problems	3
1.5	Objective, Methodology and Road Map	5
2	Taxonomy and Evaluation of TCP-friendly Congestion-Control Schemes	7
2.1	Introduction	7
2.2	TCP-Friendliness	9
2.2.1	Steady state and Transient state.....	9
2.2.2	TCP-friendly Criteria	10
2.3	Taxonomy in Fairness, Aggressiveness, and Responsiveness	12
2.3.1	Fairness Strategy	12
2.3.2	Aggressiveness Strategy.....	13
2.3.3	Responsiveness Strategy	14
2.4	Fairness Evaluation	15
2.4.1	TCP-Equivalence: Artificial-losses Testing Scenario with Identical Network Conditions	15
2.4.2	TCP Equal-share: Low-multiplexing Testing Scenario with the Same Bottleneck.....	17
2.5	Evaluation on Aggressiveness and Responsiveness	19
2.5.1	TCP-Equivalence: Two-states Artificial-losses Testing Scenario with Transient Convergence.....	19
2.5.2	TCP Equal-share: Bursty-loss Testing Scenario with the Same Bottleneck.....	21
2.6	Related Work	23
2.7	Summary	24
3	A Fast-Converging TCP-Equivalent Window-Averaging Rate Control Scheme	26
3.1	Introduction	26
3.2	Window-Averaging Rate Control (WARC)	28
3.2.1	Basic Rate-control Mechanism	28
3.2.2	Complemental Rate-control Mechanisms	30
3.2.3	Pseudo Code.....	33
3.3	Analysis on Fairness	34
3.4	Analyses on Smoothness, Aggressiveness and Responsiveness	35
3.4.1	Smoothness.....	36
3.4.2	Aggressiveness.....	37
3.4.3	Responsiveness.....	38
3.4.4	False Positive of the HR procedure.....	40
3.5	Simulation Results	42
3.5.1	Fairness.....	42
3.5.2	Smoothness.....	47
3.5.3	Aggressiveness and Responsiveness.....	48
3.6	Related Work	51
3.7	Applicability	52

3.8	Summary	53
4	On Applying Fair Queuing Discipline to Schedule Requests at Access Gateway for Downlink Differential QoS	54
4.1	Introduction	54
4.2	Problems on Using Class-based Fair Queuing	56
4.2.1	The Timing for Releasing Requests	56
4.2.2	The Determination of the Next Request	57
4.2.3	The Class-based Fairness Policy	57
4.3	A Request Scheduling Scheme for User-side Gateway	58
4.3.1	Minimum-service Order Arbiter (MOA)	59
4.3.2	Window-based Rate Controller (WRC)	61
4.4	Analysis for Delay and Fairness	62
4.4.1	Short User-perceived Latency	63
4.4.2	Delay Bound	65
4.4.3	Fairness	68
4.5	Simulation Results	69
4.5.1	Topology	70
4.5.2	Weighted Fairness and Bandwidth Sharing	71
4.5.3	User-perceived Latency	72
4.5.4	User-based Weighted Fairness	73
4.5.5	Adjustment of Outstanding Responses	74
4.5.6	Effect of U^+ on Latency	75
4.6	Affection of Exceptive Traffic	75
4.7	Field Trial	78
4.8	Summary	79
5	Conclusions	81
	Appendix 1 Smoothness Level of TCP-friendly Schemes	83
	Appendix 2 Analysis on WARC	85
	Appendix 3 Unfairness of TFRCP and TEAR	90
	References	93

List of Tables

2.1.1	The premises and proper behaviors in three criteria.....	9
2.1.2	The control parameters used in each scheme.....	9
2.2	Taxonomy in fairness, aggressiveness, responsiveness strategies.....	12
2.3	Comparison on fairness, aggressive and responsive behaviors among schemes.....	25
3.1	The control parameters used in each scheme.....	42
4.1	The utilization of link under oscillating CBR traffic ($U^+=98\%$).....	77
4.2	User-perceived latency comparisons.....	79
4.3	Comparison between MSF-RS and the original Squid on CPU time.....	79

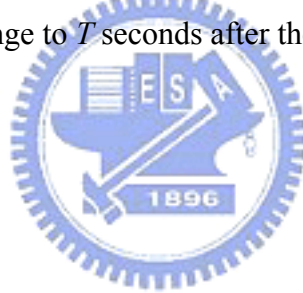


List of Figures

1.1	A typical network topology of the Internet with two types of hosts.....	1
1.2	A tree is used to organize the related works in the dissertation.....	3
1.3	A typical topology of connecting the Internet with the access link.....	5
2.1	The throughputs of TCP-friendly schemes normalized with the throughput of TCP, under the loss link whose inter-loss time has a general exponential distribution.....	16
2.2	n TCP-friendly and n TCP flows compete for the bottleneck link. The propagation delays among each set of n flows are distributed uniformly with $C/[RTT]=0\sim 0.42$	17
2.3	The comparison of the slowly convergent behaviors between TCP-friendly schemes under the two-state loss condition.....	20
2.4	The slowly aggressive behaviors of TCP-friendly schemes under the bursty-losses network.....	22
3.1	The HR procedure would be invoked when $\bar{R}_{TCP}(N) < 1/K R(t,s)$	31
3.2	The state diagram of the fluid-based timeout procedure in WARC.....	32
3.3	The pseudo code for the basic and three complementary rate control mechanisms of WARC.....	34
3.4	The smoothness effects of WARC, GAIMD, and SIMD, relative to that of TCP.....	37
3.5	The aggressiveness indices of WARC, SIMD, GAIMD and IIAD under different increasing factor m 's and the tradeoff between aggressiveness and smoothness when $m=4$	38
3.6	The responsiveness of WARC under varied decreasing factor m 's. The initial average window W before bandwidth change is 20.....	39
3.7	The responsiveness of WARC, SIMD, GAIMD and IIAD.....	40
3.8	The probability of the false-positive invocation of the HR procedure.....	42
3.9	The artificial loss-link topology is used to provide the same loss condition for any two flows running through the link R_1 - R_2	43
3.10	The dumbbell topology used to test the fair sharing between TCP and TCP-friendly flows.....	43

3.11	The throughputs of TCP-friendly schemes normalized with that of TCP under different loss probabilities.	44
3.12	The throughputs of the TCP-friendly schemes normalized with that of TCP under the artificial-loss links with different $CV[T]$	45
3.13	The competing results between TCP and five TCP-friendly schemes under the links managed by Drop-Tail are shown.	46
3.14	The competing results between TCP and five schemes under the links managed by RED are shown.	47
3.15	The smoothness of each scheme over different time scales.	48
3.16	The topology with oscillating CBR background traffic, used to test TCP-friendly schemes in terms of aggressiveness and responsiveness.	49
3.17	The comparison between five TCP-friendly schemes on aggressiveness and responsiveness under the on/off CBR background traffic	49
3.18	The number of losses encountered by WARC, WARC without HR, and other four schemes between the 600 th ~ 620 th second are plotted, which are normalized with that by TCP.	50
3.19	The normalized throughputs of TCP-friendly schemes under an oscillating CBR traffic.	50
4.1	A typical network topology that an enterprise accesses the Internet through ISP.	54
4.2	The internal architecture of MSF-RS.	59
4.3	Two procedures in <i>MOA</i> : request selector and request receiver.	61
4.4	Procedure of window-based service-rate controller (WRC).	62
4.5	The ratio on T_a of a MSF-RS gateway to an ordinary gateway.	65
4.6	The downlink can be conceptually divided into multiple sub-links.	65
4.7	The difference of the service between Class i and Class j	69
4.8	Simulation topology for three classes with service ratio 4:2:1	70
4.9	The average throughput of three classes over the four phases.	71
4.10	The average throughput of three classes over the four phases under DRR.	72
4.11	User-perceived latency comparison by decomposing time factors: queuing time and transmission time	73
4.12	The difference on the bandwidth allocated for the high-class host between	

	the host-based and class-based weighted fairness.	74
4.13	The size of W^+ is fixed in the period with insufficient traffic (the 500 th ~1000 th seconds)	74
4.14	The user-perceived latency, queuing time, and the number of packets queued in ISP-side router under different U^+	75
4.15	Two potential integrated architectures for handling the network when the uplink is a bottleneck. The diamond C represents a request classifier.....	76
4.16	Fast-responsive W^+ and full utilization of access link under oscillate CBR traffic.....	77
4.17	The test bed for field trial in the Internet	78
A1.1	The smooth level on the throughput of each scheme, relative to that of TCP, under different levels of variant-losses condition	83
A2.1	The increase curves of WARC, GAIMD, and TCP	85
A3.2	The decreasing of the CWNDs and the mean CWNDs of a TCP flow when the inter-loss time change to T seconds after the time $m \times T$	88



Chapter 1

Introduction

1.1 Bottlenecks for the Internet Traffic

The Internet traffic may encounter bottleneck at any one link. Fig. 1.1 shows a common network topology to classify the positions of bottleneck links. The first position is the link between any two edge routers (ERs) within the Internet while the second position is the access link between an Intranet and the Internet. The link in the first position is shared for public and thus has traffic coming from numerous hosts. These hosts would compete for the bottleneck with uncertain hosts. For example, Host S_1 will not know whether its packets for D_1 are competing for the link between ISP_1 and ISP_2 with packets from S_2 or other hosts. However, the link in the second position is only used by a group of users and has traffic for specific hosts. For example, the traffic passing through the link of G and EG would be associated with $H_1 \sim H_n$ only. Different from S_1 and S_2 which compete for the bandwidth of a public link, these hosts are competed for that of a private link, which is rented and managed by them.

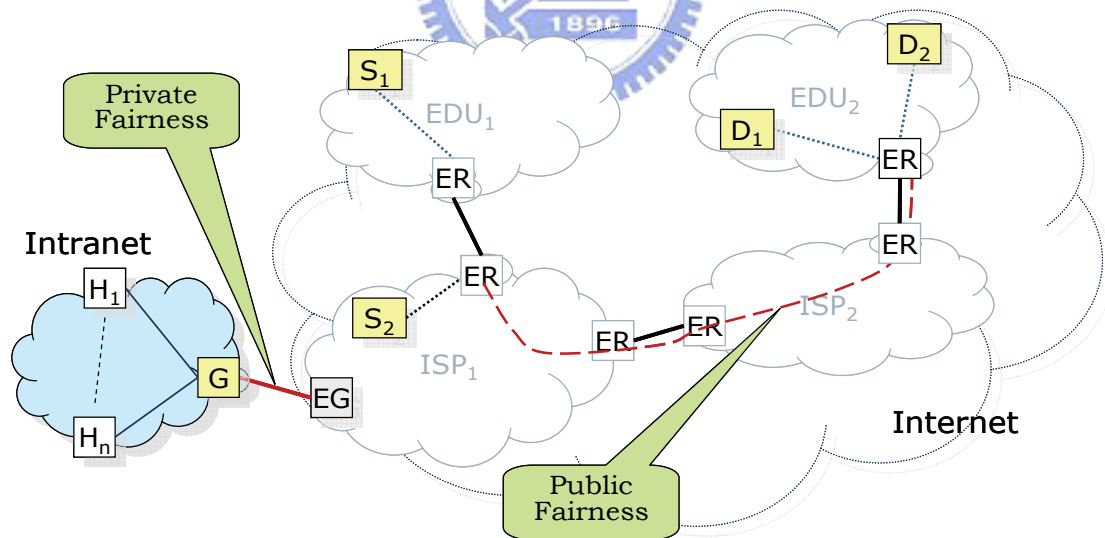


Fig. 1.1. Bottlenecks may be the links in the Internet or between the Internet and an intranet.

1.2 Public Fairness Control for TCP-Friendliness

When the bottleneck link is located in the Internet, it is important for a host to ensure that its flow fair shares the bottleneck bandwidth with other competency Internet flows, which is called *public fairness* in this dissertation. However, there is

no central mechanism in the Internet to keep the public fairness, i.e. to tell a host about how much bandwidth it should use in transmitting packets to avoid from starving other flows. In fact, the current public fairness of the Internet depends on using the same end-to-end rate control mechanism among all hosts. That is, the rates of most Internet traffic are controlled by the additive-increase/multiple-decrease (AIMD) embedded in TCP. AIMD increases the transmission rate per round-trip time (RTT) to detect and use the available bandwidth and then decreases the rate to avoid from the congestion when packet losses occur.

Unfortunately, the behavior of rate controlled by AIMD cannot satisfy the real-time streaming traffic, because such traffic prefers a smooth rate but the rate controlled by AIMD changes abruptly and largely. Therefore, new rate control mechanisms are required and their controlled traffic may coexist with that of AIMD in the Internet. In order to keep the public fairness of the Internet, the concept “TCP-compatibility”, i.e. TCP-friendliness, is suggested in RFC 2309 [BCC98]. The concept asks these rate control mechanisms to use no more bandwidth than TCP under the same network conditions such as loss ratio and RTT.

1.3 Private Fairness Control for Weighted Fairness

Compared to the public bandwidth in the Internet, the bandwidth on the access link is private. If the owner of the access link has multiple hosts commonly sharing the access link, then the owner may allocate the bandwidth for each host by his or her preference policies, which is called *private fairness* in this dissertation. Class-based weighted fairness [FJ95, PG93] is one of the policies widely used for allocating private bandwidth. By the policy, traffic running through the access link will be classified into multiple classes and each class is assigned a weight. Then, the ratio of bandwidth of any two classes would match the ratio of their weights. The class with a large weight, i.e. the high class, can get more bandwidth than that with small weight. Besides, if the high class is idle, i.e. has no traffic for transmission, its bandwidth will be proportionally allocated for other non-idle classes.

To carry out the class-based weighted fairness policy, deploying a classifier and a scheduler at the gateway is necessary. As shown in Fig. 1.1, the access link has two gateways, the user-side gateway G and the ISP-side gateway EG. G is more preferable

by user than EG to deploy these mechanisms, because the gateway G is owned by the user renting the link and is easy to manage, while EG is owned by ISP. Moreover, if a classifier is deployed at EG, it cannot classify packets by their IP addresses because the source address of all uplink packets may be identical at EG, as well as the destination address of all downlink packets may be. For hosts in the intranet, commonly sharing a public IP address for connecting the Internet is often seen because of security concern and the short of IPv4 addresses.

1.4 Related Work and Potential Problems

Fig. 1.2 shows a tree to organize the related work of this dissertation. The dissertation focuses on the issues about bandwidth fairness and divides them into the public and private fairness. Public fairness is promoted in [FF99, BCC98], which asks the Internet traffic to be transmitted by a TCP-compatible rate control scheme. Meanwhile, in order to understand what the TCP-compatible rate is, the throughput models of TCP [PFT98, AAB05] are proposed and the network conditions in the Internet [ZDP01] are investigated. Next, to implement public fairness, many TCP-compatible rate control schemes are proposed [YL00, PHP00, ROY00, JGM03, BB01, PKT99, WDM01]. These schemes intend to control the flow to have a smoother rate while using the same bandwidth as TCP. However, several literatures reveal that these schemes may have lower bandwidth under some testing cases [BBF01, LT03, VB05]. Although such schemes still confirms to “TCP-compatibility”, they are not favorable to carry the streaming traffic because they provide less bandwidth than TCP.

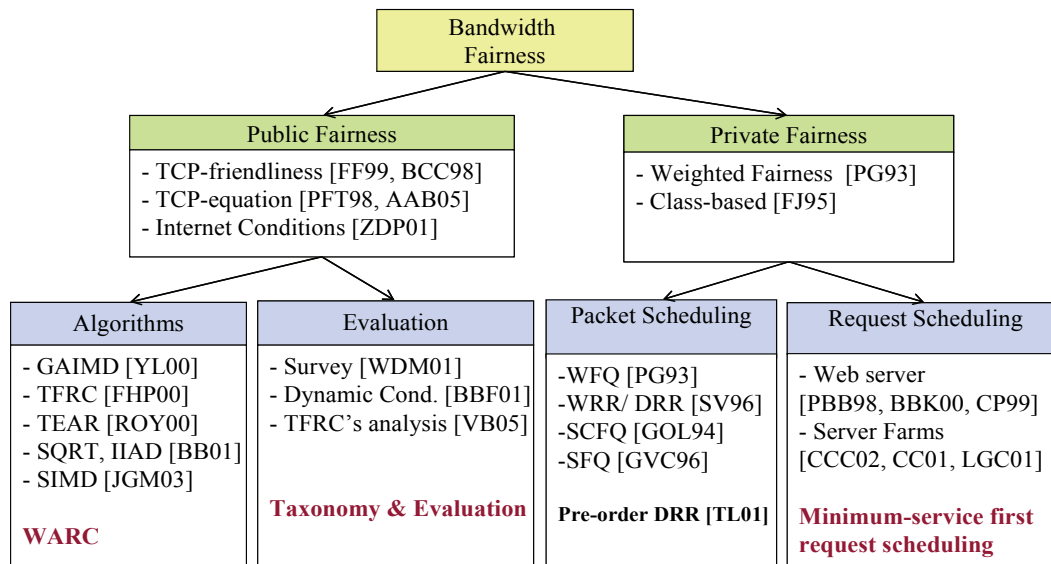


Fig. 1.2. A tree is used to organize the related work in the dissertation

Different from the public fairness which promotes each flow to use no more bandwidth than a TCP flow, the private fairness allows the differential allocations on bandwidth among the competitory flows. Class-based weighted fairness [FJ95, PG93] is such a goal and often pursued by the private fairness controls, such as WFQ [PG93], DRR [SV96], SCFQ [GOL94] and SFQ [GVC96]. Although all these schemes achieve the goal, they provide different degrees of packet latency and short-term fairness with different per-packet processing complexity. For example, DRR [SV96] is an scheduling algorithm which has $O(1)$ complexity but a little worse degree on latency and short-term fairness. Thus, pre-order DRR is proposed in our previous work [TL01] to shorten the packet latency in DRR while retaining the $O(1)$ complexity.

Unfortunately, all these packet-level scheduling schemes cannot allocate the downlink bandwidth at the user-side gateway. As shown in Fig. 1.3, when the downlink is the bottleneck, the inbound packets will queue at the ISP edge gateway. Thus, scheduling packets at the user-side gateway is useless since packets have passed through the bottleneck. For this problem, an idea is scheduling uplink requests to control the returned downlink responses since the Internet traffic most follows the request/response model. Request scheduling was used in several studies to provide differential Web QoS for different-classes users [CKD02]. These studies provided QoS services by designing request scheduling at a single Web server [PBB98, BBK00, CP99] or a web-side gateway, i.e. a gateway ahead close to a group of Web servers [CCC02, CC01, LGC01]. No published studies discussed how to design request

scheduling at the access gateway. The key difference between the previous works and this work is that the target Web servers in the former are specific and their status can be detected or controlled. The resources each request will cost could be measured in advance to assist in the scheduling mechanism. However, the servers in the latter are infinite, distributed over the Internet, and cannot be managed. It is impossible to measure the costing resources for all requests in advance.

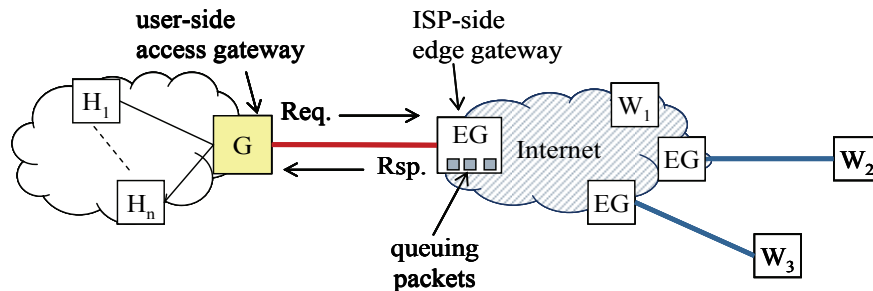


Fig. 1.3. A typical topology of connecting the Internet with the access link

1.5 Objective, Methodology and Road Map

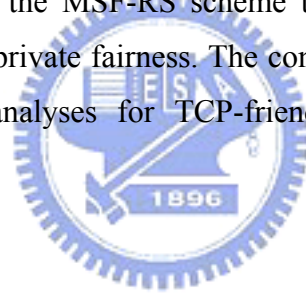
The objective of this dissertation is to propose the fairness control schemes respectively to solve the public and private unfair problems which currently existing solutions cannot handle at the end host or at the user-side gateway.

In public fairness, to clarify why the existing schemes cannot always have the same bandwidth as TCP, an investigation for eight well-know schemes is given in the dissertation. These schemes are classified and evaluated according to their underlying policies in three aspects, namely fairness, aggressiveness and responsiveness, as defined in Chapter 2. Next, according to the investigation, a fast-converging window-averaging rate control (WARC) scheme is proposed to have equal bandwidth to TCP more closely than existing schemes [YL00, PHP00, ROY00, JGM03, BB01, PKT99, WDM01]. WARC takes short time to converge its rate toward TCP's whenever the available bandwidth drastically increases or decreases. Besides, when the available bandwidth keeps stationary, WARC is the first scheme providing the same bandwidth as TCP *under any distributions* of inter-loss time. Existing schemes provide it only under some specific assumptions, e.g. the packet losses occur periodically or with a fixed probability, but these assumptions may not be realistic in the Internet [ZDP01].

In private fairness, to realize the idea of managing the downlink bandwidth by scheduling uplink requests, the dissertation first investigates the possibility of

applying the class-based fair-queuing discipline, which is widely and maturely used in scheduling packets, to schedule requests. However, we found that simply applying the discipline to schedule requests would encounter three problems. The first two are on the timing of releasing requests and the selection of the next released request, respectively. The last one is about the class-based policy, which may not suit for the user-level differentiation, i.e. may not guarantee high-class users to get more bandwidth than low-class one when more users appear in the high class. Next, based on the above investigation, we propose a minimum-service first request scheduling (MSF-RS) scheme to provide bandwidth sharing and user-based weighted fairness, i.e. a policy that guarantees the ratio of the bandwidth allocated for each high-class user to that for each low-class user matches the ratio of their weights.

The road map of the dissertation is organized as follows. Chapter 2 presents a taxonomy and evaluation for eight TCP-friendly rate control schemes. Chapter 3 proposes the fast-converging window-averaging rate control scheme for public fairness. Chapter 4 proposes the MSF-RS scheme to manage the downlink at the user-side access gateway for private fairness. The conclusions are given in Chapter 5 and advanced mathematic analyses for TCP-friendly schemes are described in Appendices.



Chapter 2

Taxonomy and Evaluation of TCP-Friendly Congestion-Control Schemes

2.1 Introduction

Real-time streaming media, such as video/audio conversations and movies online, are now often transmitted over the Internet. Because the available bandwidth in the Internet is dynamic, a congestion control mechanism is needed to prevent the media flow from suffering serious packet losses. A flow carried over TCP generally is subject to such a congestion control mechanism. TCP is the most widely-used transport protocol in the Internet, and embeds an Additive-Increase and Multiplicative-Decrease (AIMD) congestion control mechanism.

The throughput controlled by AIMD in TCP changes dramatically and frequently, which may not satisfy real-time streaming media. Many AIMD-variant and other-style congestion control schemes have been proposed to solve this problem [YL00, PHP00, ROY00, JGM03, BB01, PKT99, WDM01]. Besides being smooth, these schemes have been suggested to be TCP-friendly [BCC98] because their controlled traffic is expected to coexist with TCP traffic in the Internet. “TCP-friendly” is a generic term describing that a scheme aims to use no more bandwidth than TCP. This study discusses in detail the proper behaviors of a TCP-friendly scheme in view of the following three criteria: TCP-compatibility, TCP-equivalence and TCP equal-share.

TCP-compatibility is defined in RFC 2309 [BCC98], which says that a TCP-compatible flow, in the steady state, should use no more bandwidth than a TCP flow under comparable conditions such as packet loss rate and round-trip time (RTT), where RTT means the time required for a packet to travel from the source to the destination and back. However, a TCP-compatible congestion control scheme is not preferred if it always offers far lower throughput than a TCP flow. Hence, a better

congestion control scheme has to not only meet TCP-compatibility, but also pursue *TCP-equivalence*. A TCP-equivalent flow has the same throughput as a TCP flow if it experiences identical network conditions, which mean the same patterns of packet loss occurrences and RTT changes. Most present schemes tend to provide TCP-equivalence, rather than just TCP-compatibility. However, TCP-equivalence in *all* network conditions is hard to achieve. Various studies have described schemes that achieve compatibility without always achieving equivalence [YL00, FHP00, ROY00, BBF01, LT03, VB05].

Although a TCP-equivalent scheme consumes TCP-equivalent bandwidth when working by itself, it may not coexist well with TCP in the Internet. A TCP-equivalent scheme merely ensures the same throughput between TCP and TCP-equivalent flows when both experience identical conditions, but not that when both compete for the same bottleneck, which is exactly the real situation in the Internet. Competing for the same bottleneck does not imply experiencing identical network conditions [VB05]. Therefore, this study defines a new criterion, namely *TCP equal-share*. This criterion is more realistic than TCP-equivalence, because the most important concern is whether flows with different controls can co-exist and equally share bandwidth in the same bottleneck while coexistence is not in the picture of TCP-equivalence. Moreover, TCP equal-share is also more challenging than TCP-equivalence because a TCP equivalent flow may not be TCP equal-share, but vice versa is true.

This study has three objectives. The first objective is to be the guide for selecting from existing TCP-friendly schemes, based on the proposed taxonomy and evaluation. The second objective is to indicate the potential fault cases and causes of the eight schemes evaluated, thus helping designers to realize what needs to be enhanced. The third objective is to recommend policies for designing an ideal scheme to meet all TCP-friendly criteria. Unlike the survey of Widmer *et al.* [WDM01] which compares the functionality of various schemes, this study tests the selected schemes for the TCP-friendly criteria. Contrary to Bansal *et al.* [BBF01] who compare the transient behaviors of various schemes, this study additionally investigates these schemes under the steady state to reveal that they may use bandwidth unequal to TCP even in this case. Besides, this study investigates the bandwidth sharing between TCP and TCP-friendly flows (inter-fairness), differing from Tsaoussidis *et al.* [TZ05] who study the bandwidth sharing among a group of homogeneous flows (intra-fairness).

For TCP-friendly schemes, Table 2.1.1 summarizes the proper behaviors for the

three TCP-friendly criteria in three aspects, namely fairness, aggressiveness and responsiveness, as explained further in Chapter 2.2. Also, Table 2.1.2 shows the eight typical TCP-friendly schemes selected for this study. In Chapter 2.3, the behaviors of these schemes are classified according to their key operational characteristics to realize how they meet the three criteria. In Chapter 2.4 and 2.5, the evaluation results verify whether these schemes meet the criteria, and also reveal some further issues. Next, related work is discussed in Chapter 2.6. Finally, we make recommendations about the preferred schemes and policies, based on the observed results, in Chapter 2.7.

Notably, although TFRC is simply the predecessor of TFRC, it is selected in this study due to its simplicity, which may be preferred by the programmers of real-time applications. Moreover, Bansal *et al.* [BBF01] defined a TCP-equivalent scheme differently from this study, as a scheme with the same AIMD as TCP, but without packet loss recovery or fast retransmission.

TABLE 2.1.1. THE PREMISES AND PROPER BEHAVIORS IN THREE CRITERIA

Criterion	Network premise	Proper behaviors of a scheme		
		Steady state	Transient state	
		Fairness	Aggressiveness	Responsiveness
TCP-compatibility	Comparable conditions	Less bw	Don't care	As fast as TCP
TCP-equivalence	Identical conditions	Equal bw	As fast as TCP	
TCP equal-share	Same bottleneck			

TABLE 2.1.2. THE CONTROL PARAMETERS USED IN EACH SCHEME

SCHEME	FULL NAME	PARAMETERS	REF.
GAIMD	General additive inc./multiplicative-dec.	$\alpha=0.2, \beta=0.125$	[YL00]
IIAD	Inverse-inc./additive-dec.	$\alpha=1.0, \beta=0.67, k=1, l=0$	[BB01]
SQRT	Square-root inc./dec.	$\alpha=1.0, \beta=0.67, k=0.5, l=0.5$	[BB01]
SIMD	Square-inc./multiplicative-dec.	$\beta=0.0625, k=-0.5, l=1$	[JGM03]
AIAD/H	Additive inc./dec. with history	$\beta=0.25, k=0, l=0$	[JGM03]
TFRC	TCP-friendly rate control protocol	Interval=5 seconds	[PKT99]
TFRC	TCP-friendly rate control	The number of samples=8	[FHP00]
TEAR	TCP-emulation at receiver	The number of samples=8	[ROY00]

2.2 TCP-friendliness

2.2.1 Steady state and Transient state

As shown in Table 2.1.1, the term “steady-state” is used in the description of the three criteria. A steady-state network originally means that a network with a negligible change over an arbitrarily long period. By this definition, the Internet would not be in the steady-state condition unless the term “arbitrarily long” is removed from the definition. The measured result in [ZDP01] reveals that the packet loss condition experienced by an Internet flow may consist of multiple *minute-scale* steady-state regions, and the time interval between any two consecutive losses may be mutually independent and have the same probability distribution, i.e. be independently and identically distributed (i.i.d.), within a region. Thus, a TCP-friendly scheme should use the same bandwidth as TCP in a steady-state region, while being *aggressive* enough to capture the available bandwidth and being *responsive* enough to protect itself from congestion, as the packet loss condition changes across regions (the transient state). Notably, a packet loss (event) in this study denotes an event causing a TCP flow halving its congestion window. Such an event may imply that multiple consequent packets are discarded. For convenience, this study, like other studies [YL00, FHP00, ROY00, BBF01, LT03, VB05], ignores the term “event”.

2.2.2 TCP-friendly Criteria

This study uses the following three criteria to describe the proper behaviors of a TCP-friendly scheme. Vojnovic *et al.* presented a criterion, named “conservative” [VB05]. However, this criterion is suitable only for evaluating schemes that use TCP throughput formula, and therefore is not considered herein.

1) TCP-compatibility: The basic criterion, introduced in RFC 2309 [BCC98], is defined as, “A TCP-compatible flow is responsive to congestion notification, and uses no more bandwidth in the steady state than a conformant TCP flow running under comparable conditions (e.g. packet loss rate, RTT).” As shown in Table I-A, this criterion forbids a scheme from providing a flow with more bandwidth than TCP, in order to protect TCP flows from starvation. Based on this definition, a TCP-compatible flow should decrease the throughput at least *as fast as* TCP when the packet loss condition becomes severe, i.e. responsive but not necessarily aggressive. Otherwise, the compatibility criterion would be violated during the long convergence time of the flow.

2) TCP-equivalence: This study defines the criterion as, “If given identical network conditions, then a TCP-equivalent flow uses the same bandwidth as a TCP flow when the network condition is either in the steady or transient state.” This

criterion, unlike “TCP-compatibility”, requires the same bandwidth, not just “no more” bandwidth than TCP. Therefore, a TCP-equivalent scheme is more desirable for transmitting media traffic, because it provides more bandwidth than a TCP-compatible scheme. Moreover, to meet the criterion in the transient state, a TCP-equivalent scheme must consider aggressiveness besides responsiveness. That is, if more bandwidth becomes available, then a TCP-equivalent scheme should increase the throughput of its controlled flow as fast as TCP. Finally, TCP-equivalence requires “identical network conditions”, rather than “comparable conditions”, to ensure the same patterns of packet loss occurrences and RTT changes. The requirement is necessary for testing a scheme whether to have the same throughput as TCP, because TCP has different throughputs under the same mean but different variances of loss rate or RTT [AAB05].

A TCP-equivalent scheme may work well in routers which use well-designed Active Queuing Management (AQM) algorithms to manage their bottleneck links, because such routers may offer the needed *premise*, namely “given identical network conditions” to TCP and TCP-equivalent flows. However, if this premise is not supported, then a TCP-equivalent flow may have *more* throughput than a TCP flow when the TCP-equivalent flow experiences fewer packet losses from the routers. To support the premise, these AQMs apply equal packet loss rate on flows of the same throughput, with the loss rate being directly proportional to the throughput. Since TCP and TCP-equivalent flows adjust the throughput based on their loss rates regulated by the AQM, they finally would have the same throughput and loss rate. Readers interested to this issue may refer to Gwyn *et al.* [CLB04].

3) TCP equal-share: This study defines the criterion as, “A TCP equal-share flow uses the same bandwidth as a TCP flow if both flows compete for the same bottleneck.” This criterion should hold regardless of whether the network conditions experienced by the two flows are identical. This criterion differs from TCP-equivalence in its premise, “competing for the same bottleneck”, which implies “*competing for the shared bandwidth resources*”, but it is not necessary for TCP-equivalence.

TCP equal-share is more realistic than TCP-equivalence. A new scheme is safe to deploy if it provides the same bandwidth as TCP when competing for the same bottleneck, not just when it has identical network conditions. However, achieving TCP equal-share is more challenging than achieving TCP-equivalence, because

competing for the same bottleneck does not imply experiencing identical network conditions [ZDP01]. Therefore, a TCP-equivalent flow may not be TCP equal-share if it experiences different network conditions from a TCP flow. However, a TCP equal-share flow should have the same bandwidth as a TCP flow, regardless of network conditions, implying that it is also TCP-equivalent.

2.3 Taxonomy in Fairness, Aggressiveness, and Responsiveness

The section investigates the fairness, aggressiveness and responsiveness policies taken by the selected schemes, as summarized in Table 2.2.

TABLE 2.2. TAXONOMY IN
FAIRNESS, AGGRESSIVENESS, RESPONSIVENESS STRATEGIES

Policy	<i>Fairness</i>	<i>Aggressiveness</i>		<i>Responsiveness</i>
Aspect	<i>throughput adjusting</i>	<i>step of each inc.</i>	<i>curve type</i>	<i>life cycle of loss statistics</i>
GAIMD	Window-based	Non-historical	Linear	Variable-history
IIAD	Window-based	Historical	Sub-linear	Non-historical
SQRT	Window-based	Historical	Sub-linear	Variable-history
SIMD	Window-based	Historical	Super-linear	Variable-history
AIAD/H	Window-based	Historical	Linear	Non-historical
TFRC	Rate-based	Non-historical	Super-linear	Fixed-history
TFRC	Rate-based	Historical	Linear	Fixed-history
TEAR	Rate-based	Historical	Linear	Fixed-history

2.3.1 Fairness Strategy

The fairness policy of a scheme describes how the scheme adjusts a flow to have equivalent throughput to a TCP flow in the long term under the steady state. As shown in Table 2.2, the selected schemes use two fairness policies, window-based (WB) and rate-based (RB).

The WB fairness policy controls the throughput by adjusting the congestion window (CWND). CWND represents the number of packets that can be freely sent without waiting for their acknowledgements, and is updated by a set of control

parameters, see Table 2.1.2. A specific relationship exists between the parameters, giving a scheme equal throughput to the TCP. Applying this policy requires the developments of control parameters and their specific relationship. For instance, GAIMD uses two parameters, α and β , to control its CWND, increasing CWND by α for every RTT and decreasing CWND by β if a packet loss occurs. A specific relationship $\alpha=3\beta/(2-\beta)$ exists between α and β for achieving the same throughput as TCP. Five of the selected schemes, GAIMD, SQRT, IIAD, SIMD and AIAD/H, apply the WB policy.

The RB fairness policy directly adjusts the throughput by finely controlling the time between sending two packets and thus has a smoother rate than the WB policy. The RB policy continues to estimate the potential throughput of a TCP flow during its lifetime and repeatedly adjusts the sending rate according to this estimated TCP throughput, enabling a flow to have equal throughput to TCP. Applying this policy requires the developments of scheme for estimating the TCP throughput and determining when to adjust the sending rate. The RB policy is applied in three schemes, TFRCP, TFRC and TEAR.

2.3.2 Aggressiveness Strategy

The aggressiveness policy of a scheme describes how the scheme increases the throughput of a flow before encountering the next packet loss. As shown in Table 2.2, the non-historical policy is taken by GAIMD and TFRCP. The step of increase is *independent of the history of packet losses*, and is thus fixed during the whole life of the flow. Unfortunately, this behavior brings the tradeoff between aggressiveness and smoothness. For instance, when GAIMD employs a small step for smoothness, a slow rate of increase may prohibit GAIMD from achieving either TCP-equivalence or TCP equal-share when the loss condition changes dramatically. Conversely, TFRCP doubles its rate if it does not encounter any loss during a fixed time interval, which makes it super-linear, i.e. fast and aggressive, but possibly causes large oscillation, i.e. poor smoothness.

By contrast, the historical policy has a variable step. For example, to achieve smoothness, SIMD initially takes a smaller increasing step than TCP after encountering a packet loss. SIMD then enlarges the step according to the historical maximum CWND, to increase the aggressiveness before encountering the next loss. The historical policy also enables AIAD/H to dynamically determine a step for

linearly increasing the throughput. AIAD/H seems to be more adaptive than GAIMD.

Three of the schemes with the historical policy, namely SQRT, IIAD and SIMD, have non-linearly increasing curves between packet losses, because they change their steps per RTT, instead of per loss. SQRT and IIAD have sub-linearly increasing curves, because they shorten the step inversely with increasing \sqrt{CWND} and CWND, respectively. By contrast, SIMD has a super-linear behavior and thus has the fastest increasing rate because the step in SIMD is enlarged with the time escaped from the latest loss.

2.3.3 Responsiveness Strategy

The responsiveness policy of a scheme describes how the scheme decreases the throughput of a flow when the packet loss condition becomes severe. The key difference among the policies is the life cycle of the loss statistics used in adjusting the new throughput. The loss statistics include the number of inter-loss packets (the received packets between two losses), the inter-loss time, or the loss rate measured in an interval. There are three policies, namely non-historical, fixed-history and variable-history, as shown in Table 2.2.

The non-historical policy ignores the historical packet loss statistics in decreasing throughput, and thus decreases the throughput at a *constant* speed, thus producing a tradeoff between responsiveness and smoothness. For example, to ensure smoothness, IIAD and AIAD/H employ a small decreasing speed, leading to a long convergence time and the violation of all three criteria, particularly when a significant change of loss condition occurs.

The other two policies consider the historical packet loss statistics in order to decrease the throughput. In the variable-history policy, loss statistics of a large value may have a longer duration to affect the throughput than that of a small value. For example, CWND in GAIMD controls the throughput, and can be regarded as a weighted average over all historical values on inter-loss time, where the values obtained earlier have smaller weights [ABB05]. Therefore, early but large values still affect the throughput, even when their weights are small. However, schemes with fixed history only consider the latest n loss statistics when computing the new throughput. A loss statistic, regardless of its value, is eliminated from the computation if it is not among the latest n values. Fixed-history schemes include TFRC, TEAR and TFRCP.

2.4 Fairness Evaluation

We use ns-2 simulation [NS06] and examine the fairness of eight different schemes to determine whether they meet the TCP-equivalence and TCP equal-share criteria. The source codes of TEAR, TFRC, SIMD, and AIAD are not included in the package of ns-2 simulation, but instead are published individually on the Web sites of their authors. Also, this study like [FHP00, JGM03, BB01] uses SACK [MMF96] as the TCP version and assumes no delayed acknowledgments. For the simulation, we use packets that were 1,000 bytes long and a maximum window size of 200 packets.

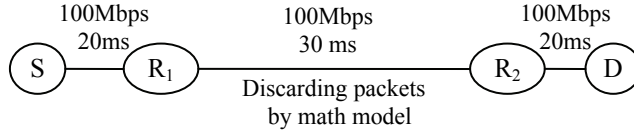
2.4.1 TCP-equivalence: Artificial-losses testing scenario with identical network conditions

A link with artificial packet losses was used to test for TCP-equivalence. The link discards the passing packets with a specific mathematical model. Such a link guarantees that any two passing flows experience identical loss conditions, thus satisfying the premise in TCP-equivalence, making this link suitable for the test of TCP-equivalence. Sufficient bandwidth was allocated for this link to prevent the packets from being dropped due to overflow.

The selected schemes were tested to determine whether they are robust enough to have the same throughput as TCP under varied artificial links, which have different means or Coefficient-of-Variations (CVs) of inter-loss time. The two statistics were varied because both affect the TCP throughput [AAB05]. A general exponential random variable allows its coefficient-of-variation to be changed while fixing its mean, or vice versa, so it is employed to drop packets at the link. The time between two packet losses thus forms a general exponential distribution, which is also used in [VB05] to investigate the conservativeness of TFRC. Only the testing result under links with different coefficient-of-variations is shown herein. The result with different means has already been obtained [JGM03, BB01].

The artificial link plotted as the link R_1 - R_2 in Fig. 2.1(a), drops one packet every T seconds. T denotes a general exponential distributed random variable where $E[T]$ is fixed at 5 and $CV[T]$ uniformly increases from 0 to 1. The results in Fig. 2.1 were averaged from five runs of 5200 seconds each, where the data within the first 200 seconds were discarded, and the mean coefficient-of-variation of the simulation results between the five runs was 0.025. Because this coefficient-of-variation is small,

it is ignored in plot to improve the clarity of the figure.



(a) The artificial-loss topology used in Chapter 2.4.A and 2.5.A

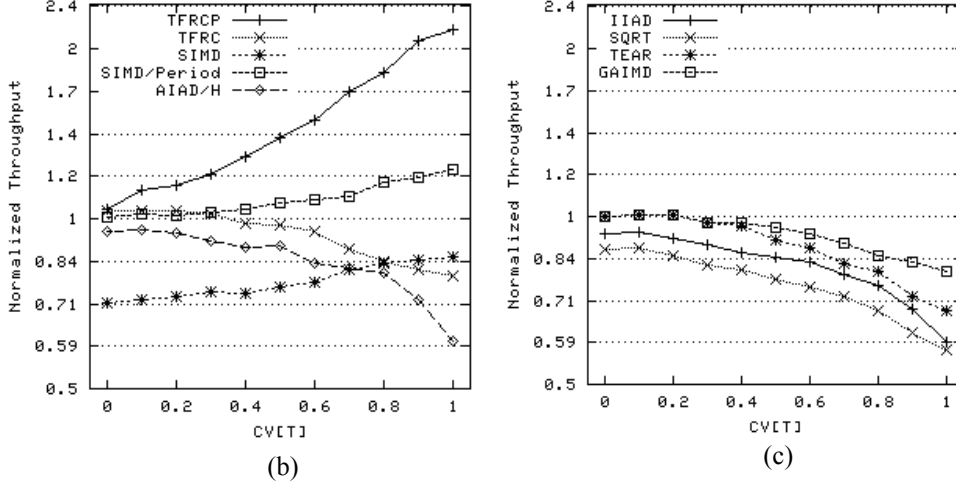


Fig. 2.1. The throughputs of TCP-friendly schemes normalized with the throughput of TCP, under the loss link whose inter-loss time has a general exponential distribution. For clarity, results are separately shown in (b) and (c).

Observation 1: Non-periodic losses should be considered in adopting WB/RB fairness policies.

Figures 2.1(b)(c) reveal that none of the WB/RB schemes meet TCP-equivalence under *non-periodic* packet loss ($CV[T]>0$). When $CV[T]=1$, GAIMD and TFRC only have 80% throughput of TCP while TEAR, IIAD, SQRT, and AIAD/H have 60% on average, because all schemes, except SIMD, were proposed based only on the periodic-loss assumption, i.e. the packet losses occur periodically. The unfairness under $CV[T]=1$ should be handled by these schemes because the inter-loss time in the Internet may approximate an i.i.d. exponential distribution equivalent to the link with $CV[T]=1$, according to the observation in [ZDP01].

Notably, the TFRCP and SIMD flows exhibit a different trend from other flows in Fig. 2.1(b)(c). The difference of TFRCP is due to the convex TCP throughput equation and the fixed rate-adjusting period [VB05], while that of SIMD occurs because its specific relationship between parameters is based on the packet loss model with $CV[T]\simeq 1$ [JGM03]. Figures 2.1(b) also plots the curve of SIMD variant, SIMD/Period, with this design based on $CV[T]=0$. Unfortunately, SIMD/Period violates the TCP-compatibility criterion under non-periodic conditions.

2.4.2 TCP equal-share: Low-multiplexing testing scenario with the same bottleneck

A dumbbell topology provides the premise of TCP equal-share, i.e. “competing for the same bottleneck,” and thus is used to verify the TCP equal-share of a scheme in the steady state. As shown in Fig. 2.2(a), n TCP-friendly flows compete with n TCP flows for a single bottlenecked link. All flows have backlogged data for the whole testing period. This study particularly investigates a low-multiplexing scenario [F00], where n is small and Drop-Tail is deployed to manage the bottleneck link, because previous results [YL00, FHP00, ROY00, BBF01, LT03] imply that a TCP-equivalent flow may violate TCP equal-share under such a scenario. Drop-Tail is a queuing management algorithm which discards new arrival packets when its managed queue is full.

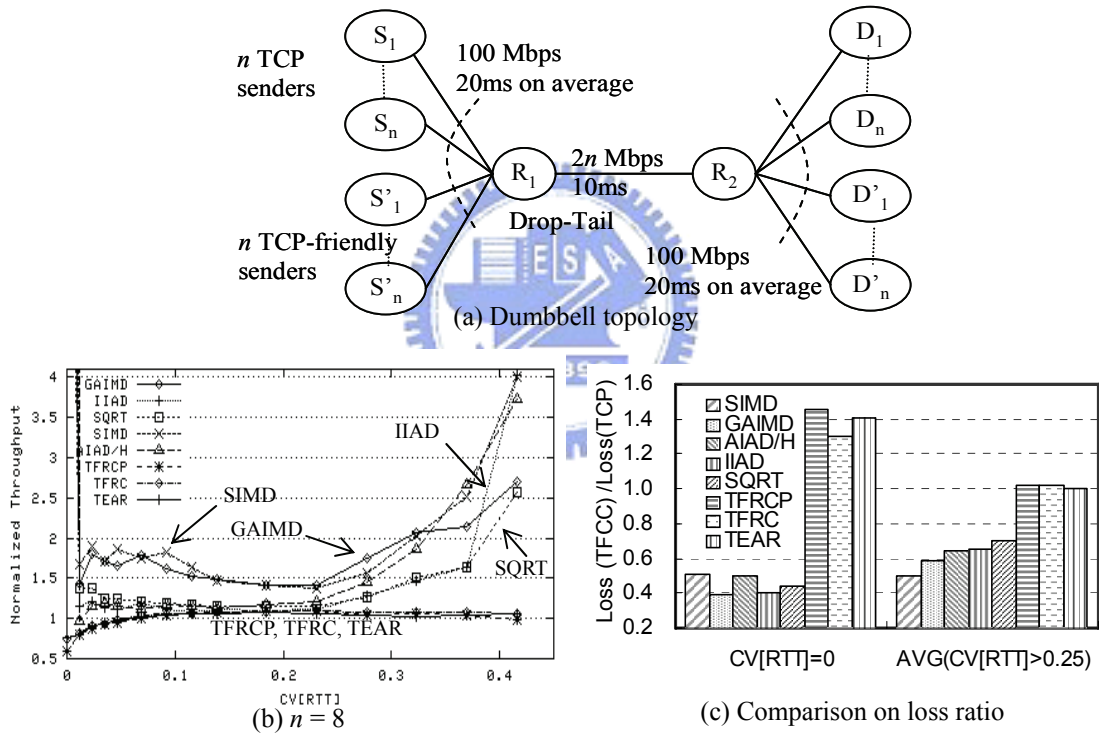


Fig. 2.2. n TCP-friendly and n TCP flows compete for the bottleneck link. The propagation delays among each set of n flows are distributed uniformly with $CV[RTT]=0\sim 0.42$.

To indicate the cause of the violation, the scenario used in [YL00, FHP00, ROY00, BBF01, LT03] was slightly modified at two points. First, instead of using a fixed capacity, e.g. 15 or 60 Mbps, the link had $2n$ Mbps. Such a link can provide on average 1Mbps of bandwidth for each flow, avoiding the influence of the TCP timeout-handling mechanism, as expected from previous studies [YL00, FHP00, ROY00, BBF01, LT03]. Second, although multiple rounds were tested for the same n ,

the RTT-heterogeneity of n TCP flows and of n studied flows were enlarged equally over different rounds. The RTT-heterogeneity of n flows represents the coefficient-of-variation of the RTTs of these flows, denoted as $CV[RTT]$. The mean end-to-end propagation delay was set to 50ms for all rounds. The queue size was 1.5 times the bandwidth-delay product.

Observation 2: RB fairness policy wins and RTT-heterogeneity matters for TCP equal-share.

Figure 2.2(b) indicates that the tested schemes do not always ensure TCP equal-share under the scenario, because they are based on the premise of TCP-equivalence, i.e. “any two flows experiencing identical network conditions,” but not that of TCP equal-share. Thus, these schemes cannot have the same throughput as TCP when the premise of TCP-equivalence is false, i.e. they do encounter different numbers of packet losses.

To show that the premise of TCP-equivalence is false under the scenario, Fig. 2.2(c) plots the normalized packet loss rate experienced by the TCP-friendly flows with the shortest RTT, compared with that of TCP flows. The loss rates of shortest-RTT flows are shown because their differences are the most significant among all flows. Three RB schemes, namely TFRCP, TFRC and TEAR, clearly suffer a higher loss rate than TCP at $CV[RTT]=0$, but an equal rate at $CV[RTT]>0.25$, which explains their bandwidth sharing with TCP in Fig. 2.2(b). Similarly, the other five schemes suffer a lower loss rate than TCP, so they occupy much more bandwidth than TCP.

Figure 2.2(b) also reveals that the *RTT-heterogeneity* of the competing flows significantly affects *the fairness* between TCP and TCP-friendly flows. GAIMD and SIMD occupy more bandwidth on average than TCP flows (1.5~4 times), particularly when $CV[RTT]=0$ (>10 times), where the number of competing flows is small ($n=8$, total is 16). The seriously unfair situation at $CV[RTT]=0$ also exists even when the total number of competency flows is 64.

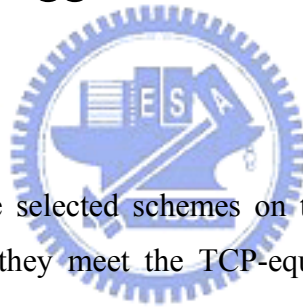
The unfair situation in the five WB schemes results from their exercising the packet acknowledgement mechanism. These schemes, like TCP, delay the transmission of the next data packet if the transmitter does not receive an ACK packet because the queue of a router in the transmission path has overflowed. By the delay, they encounter fewer packet losses and thus have higher throughput than the three RB

schemes. Moreover, because the overflow is alleviated by TCP significantly reducing its CWND, these five schemes, which slowly reduce their CWNDs, may monopolize the link until the queue is overflowing again. Thus, they have higher average throughput than TCP.

Although neither the WB and RB fairness policies can ensure TCP equal-share, the RB flows would experience similar packet loss rate to TCP flows, and can meet TCP equal-share in most cases, i.e. under $CI/[RTT]>0.05$. By contrast, the WB flows may severely starve TCP flows. Therefore, the RB fairness policy should have a better chance than WB of meeting the TCP equal-share. Notably, these TCP-friendly schemes were also tested under a topology with multiple bottlenecks, but the results reveal that their TCP equal-share is unrelated to the number of bottlenecks, when this number increases from 1 to 10.

2.5 Evaluation on Aggressiveness and

Responsiveness



This section evaluates the selected schemes on their aggressive and responsive behaviors to verify whether they meet the TCP-equivalence and TCP equal-share criteria.

2.5.1 TCP-equivalence: Two-state artificial-losses testing scenario with transient convergence

The objective of the testing is to observe whether the throughput of the schemes converge as fast as TCP. An artificial-loss link was used, as in 2.4.A, because it satisfies the premise of TCP-equivalence. However, a two-state packet loss model was adopted in the link to simulate large changes in the loss conditions. A packet was dropped every 5 seconds during the 100th~800th seconds, and every 1 second at other times. The result after 100 seconds exhibits aggressive behavior, and that after 800 seconds exhibits responsive behavior. The RTT of the testing flow was about 140ms.

Observation 3: Throughput-inversed aggressive, defined below, and non-historical responsive policies are inadequate.

Figure 2.3(a) and the left part of Fig. 2.3(b) reveal that IIAD and AIAD/H take

700 seconds to increase their throughput to the new steady throughput. Such a long time is unacceptable, particularly since the other six schemes reach steady throughput within 100 seconds. Surprisingly, although AIAD/H has a linearly increasing curve between two packet losses as mentioned earlier, it has a slower convergence than IIAD. Under this scenario, the reason that both schemes seriously violate TCP-equivalence is their slowly increasing behaviors across over multiple losses, instead of between two losses. Both schemes shorten the increasing step inversely with their throughput per loss. Herein such an unfavorable slow aggressive behavior is called a *throughput-inversed aggressiveness policy*.

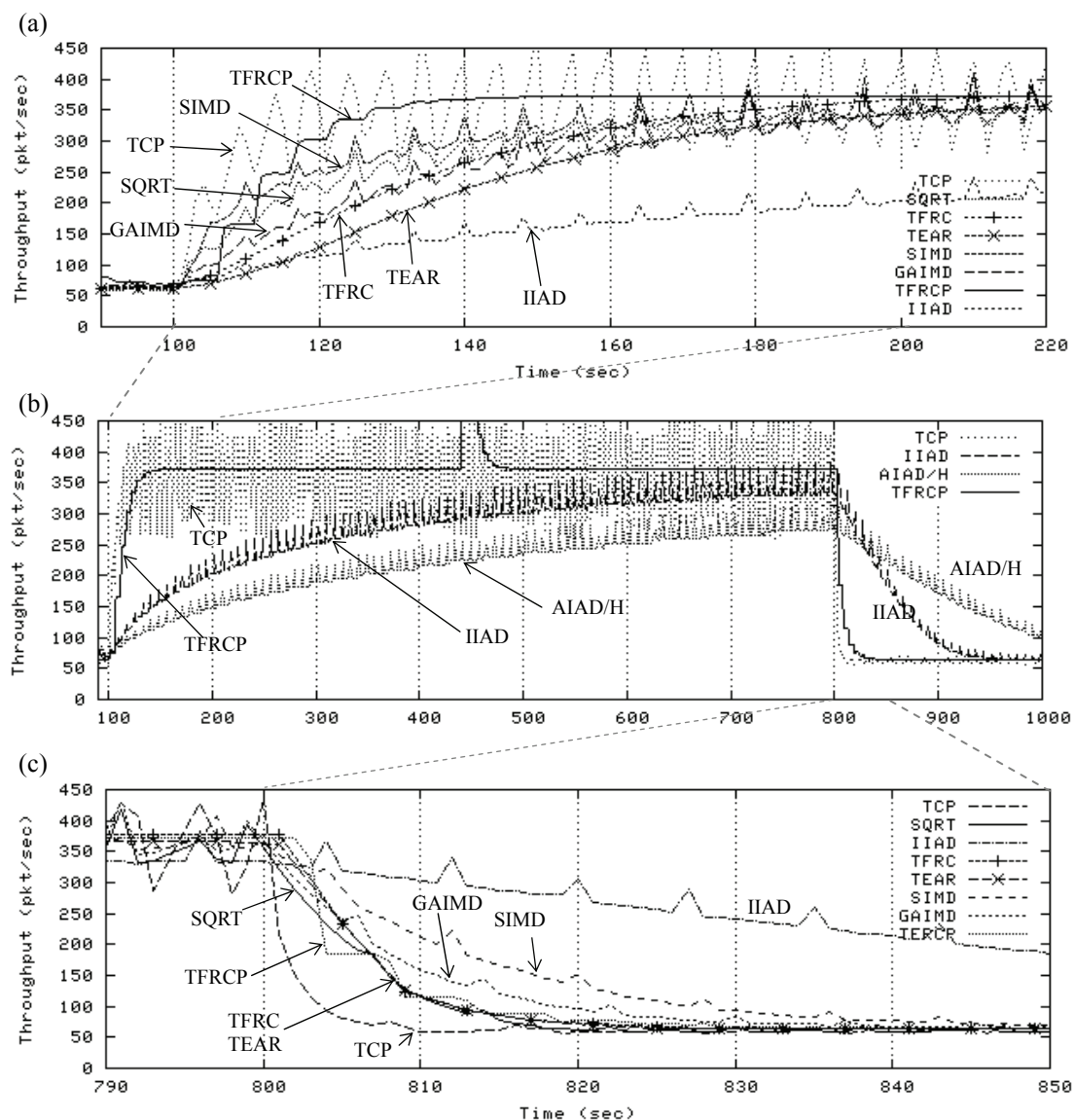


Fig. 2.3. The comparison of the slowly convergent behaviors between TCP-friendly schemes under the two-state loss condition

Figure 2.3(c) and the right part of Fig. 2.3(b) verify that the *non-historical responsiveness policy* does not satisfy the TCP-equivalence criterion. The policy

brings IIAD and AIAD/H longer convergence time than the other schemes. Figure 2.3(c) reveals that the *fixed-history policy* usually takes a shorter time to converge than the *variable-history policy*. TFRC, TFRCP, and TEAR take 20 seconds to converge, which is half the time of GAIMD and SIMD. However, the results also reveal that SQRT, which has a *variable-history policy*, also has a short convergence time. Further analysis indicates that the control parameters used in SQRT have the advantage of a short convergent time.

2.5.2 TCP equal-share: Bursty-loss testing scenario with the same bottleneck

To test whether a scheme in the transient state meets the TCP equal-share criterion, a two-state constant-bit rate (CBR) arrival traffic with obviously different rates between on and off periods was applied to the dumbbell bottleneck scenario used in Chapter 2.4.B. The oscillating CBR traffic emulates the arrival of a group of TCP flows, significantly changing the packet loss condition of the bottleneck, and thus providing the required transient-state scenarios. Such traffic in [BBF01] is used to observe how a GAIMD, TFRC, IIAD, or SQRT flow competes with a bursty arrival of TCP traffic.

Whereas Bansal *et al.* [BBF01] showed the statistical behavior for the selected schemes, this study reveals their micro behavior in one on/off period. Additionally, this study tested four schemes, SIMD, AIAD/H, TFRCP, and TEAR, which were not tested in [BBF01], are included here. The bottleneck in the test was a 15Mbps link managed with Drop-Tail, where the rate of the two-state CBR traffic oscillated between two values, 14Mbps and 9Mbps, to vary the bandwidth available for the TCP-friendly flow to 1Mbps and 6Mbps, respectively. The propagation delay of flows was 60ms, and the queue size was set to 1.5 times the bandwidth-delay product [BBF01].

Observation 4: Historical/super-linearly aggressive and fixed-history responsive policies are satisfactory.

Figure 2.4(a) indicates that *historical/super-linear aggressiveness* is the preferred policy, because it enables SIMD to use the available bandwidth as quickly as TCP, i.e., to meet the TCP equal-share criterion, and to have a smooth rate after the convergence. By contrast, as shown in Fig. 2.4(b), the non-historical/super-linear policy of TFRCP is not recommended, because a non-historical policy does not change the increasing step to a small value after the convergence, thus causing large oscillations in TFRCP.

Notably, care should be taken when using the history. AIAD/H also uses a historical aggressiveness policy, but takes too short a history to allow a stable increase during the testing time. TFRCP and AIAD/H are not recommended because of their instability. The *historical/super-linear aggressiveness* policy has the fastest rate of increase, and provides both smoothness and aggressiveness, making it most likely to meet the TCP-equivalence and TCP equal-share.

Figure 2.4(c) indicates that the *fixed-history responsiveness policy* meets TCP equal-share in terms of responsiveness *by encountering fewer packet losses* than other policies. Although all schemes reduce their throughput within about 15 seconds, Fig. 2.4(c) shows that the fixed-history schemes, such as TFRC and TEAR, encounter fewer losses during convergence than variable-history schemes, such as SIMD and GAIMD. Therefore, the *fixed-history responsiveness* policy appears to have the best chance of meeting the three criteria, because it considers *bounded* statistics and thus may reach convergence with fewer packet losses or shorter time than other policies, particularly when the loss statistics change significantly.

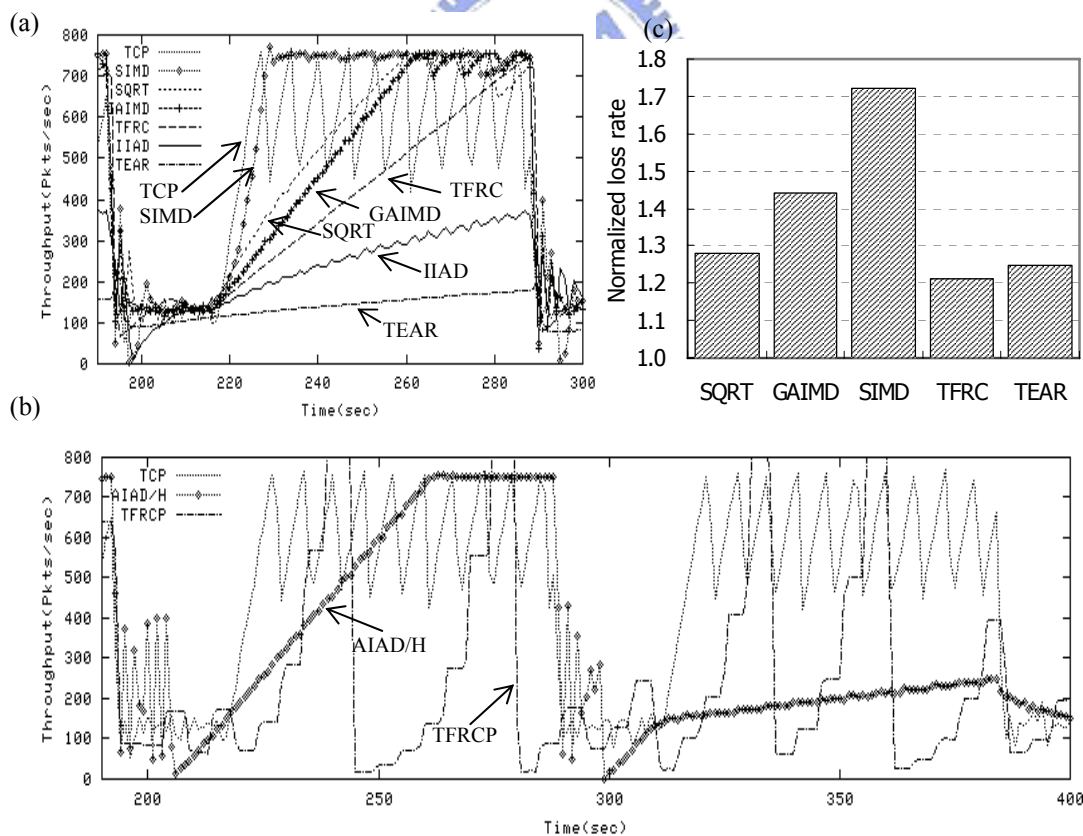


Fig. 2.4. (a) and (b) The slowly aggressive behaviors of TCP-friendly schemes under the bursty-losses network. The (b) has longer timescale than (a) to show that TFRCP and AIAD/H have different behaviors in each on/off period. (c) The number of loss events encountered by TCP-friendly schemes, normalized to that by TCP, under the low-available bandwidth case.

2.6 Related Work

Many AIMD variants have been proposed for different purposes. This study evaluates variants that aim to have a throughput smoother than but equivalent on average to TCP's. Therefore, this study does not evaluate some schemes, e.g. AIMD-FC [LT03] and [NK04], that stress fast convergence in high-speed links. Additionally, this study focuses on the inter-fairness, i.e. whether a scheme shares the same bandwidth with TCP. Intra-fairness, i.e. the fairness among the flows controlled by the same scheme, is discussed in [TZ05,G04]. Moreover, the schemes selected herein detect congestions only by packet losses as TCP Reno and SACK [MMF96] do. Actually, the RTT variation can be used for the detection, as in TCP Vegas [BP95], which also provides a smooth rate. However, RTT-based scheme may share bandwidth unfairly in the Internet where most traffic is still controlled by loss-based versions of TCP. C. Zheng and V. Tsaoussidis [ZT06] recently proposed a scheme using both packet losses and RTTs, which may be the solution to the unfairness problem.

Although the topologies discussed have appeared in the literature, e.g. [BB01,VB05], *this study revises the simulation scenarios and compares additional schemes to reveal undiscovered phenomena.* For example, this study uses a common topology – dumbbell – to investigate the TCP equal-share of the schemes, but changes the RTT-heterogeneity to display the difference between WB and RB schemes. Tsaoussidis *et al.* considered the RTT-heterogeneity in [TZ05] but for the intra-fairness of GAIMD flows. Moreover, this study like [BB01] uses the oscillating CBR traffic, but includes four extra schemes to show three interesting results, i.e. SIMD has the fastest aggressiveness, AIAD/H and TFRC are the most unstable, and TEAR has the slowest aggressiveness. Additionally, this study used the general exponential distribution, as used by Vojnovic *et al.* [VB05] who show that TFRC may have a lower throughput than TCP under non-periodic losses due to its design. However, this study reveals that schemes other than TFRC have the same unfairness phenomenon, although they control the throughput with methods different from TFRC.

Besides the congestion control, other factors must also be considered when

designing a protocol for carrying streaming traffic. E. Kohler *et al.* [KHF06] discussed these factors in depth, and proposed the Datagram Congestion Control Protocol (DCCP). DCCP allows free selection of a congestion control scheme, and therefore is the most realistic means for practical use of schemes addressed in this study. The protocol currently only includes two schemes, namely TCP-like and TFRC. We strongly encourage the addition of other schemes to the protocol.

2.7 Summary

For a TCP-friendly congestion control scheme, meeting TCP-compatibility only protects TCP flows from starvation and network from congestion, but cannot guarantee that the media flow obtains equal throughput to TCP. A good scheme should use the same throughput as TCP in the steady state, but as aggressive and responsive as TCP in the transient state. To examine whether the present TCP-friendly schemes meet the TCP-equivalence and TCP equal-share criteria, we classify the behaviors of eight typical schemes in terms of fairness, aggressiveness, and responsiveness. Additionally, we test the conformance these schemes to the criteria under four scenarios, namely non-periodic losses, low-multiplexing, two-state losses, and bursty-losses.

Table 2.3 summarizes the evaluation result for the eight selected schemes for fairness, aggressiveness and responsiveness. A comparison of the results in this table with the taxonomy results shown in Table II demonstrates that a TCP friendly scheme may have desirable TCP-equivalence and TCP equal-share in a general network condition, if it takes the rate-based fairness, historical/super-linear aggressiveness and fixed history responsiveness policies. The evaluation results of the three recommended policies are shaded in Table 2.3, which are obviously more satisfactory than that of other policies.

Unfortunately, no scheme simultaneously takes the three recommended policies for meeting the three criteria. However, if protecting TCP flows from starvation, i.e. meeting TCP-compatibility, is the major concern, then TFRC is recommended. TFRC uses the rate-based fairness and fixed-history responsiveness policies, and therefore has better behaviors under most scenarios than others on average, as shown in the row TFRC of Table 2.3. However, if fast aggressiveness is the most important property,

SIMD is recommended, since it takes the shortest time to converge and then maintains a stable throughput, due to its historical/super-linear aggressiveness policy. Nevertheless, SIMD violates TCP-compatibility under low-multiplexing bottleneck, because of its window-based fairness policy. Moreover, SIMD spends longer time or encounters more packet losses before reducing its throughput to the available bandwidth because of its variable-historical responsiveness policy.

Table 2.3. COMPARISON ON FAIRNESS, AGGRESSIVE AND RESPONSIVE BEHAVIORS AMONG SCHEMES

Behavior	Fairness			Aggressiveness		Responsiveness	
Criterion	TCP-eq (TCP-comp) ⁺	TCP eq-share		TCP-eq (TCP-comp)	TCP eq-share	TCP-eq (TCP-comp)	TCP eq-share
Scenario	Non-periodic Losses	Low-multiplexing		Two-state Losses	Bursty Losses	Two-state Losses	Bursty Losses
		Homogeneous RTTs	Heterogeneous RTTs				
GAIMD	$\Delta(O)$	X	X	$\Delta(O)$	Δ	$\Delta(\Delta)$	Δ
IIAD	X(O)	Δ	X	X(O)	X	X(X)	X
SQRT	X(O)	Δ	X	O(O)	Δ	O(O)	O
SIMD	$\Delta(O)$	X	X	O(O)	O	$\Delta(\Delta)$	X
AIAD/H	X(O)	Δ	X	X(O)	X	X(X)	X
TFRCP	X(X)	Δ	O	$\Delta(O)$	X	O(O)	O
TFRC	$\Delta(O)$	Δ	O	$\Delta(O)$	Δ	O(O)	O
TEAR	X(O)	Δ	O	X(O)	X	O(O)	O

O: satisfactory Δ : Acceptable X: Unacceptable

TCP-eq: TCP-equivalence TCP-comp: TCP-compatibility TCP eq-share: TCP equal-share

⁺ The evaluating results on TCP-compatibility are shown in the parentheses.

As a result of this study, we also observed the following: (1) a scheme should consider non-periodic loss models when taking any one of the fairness policies; (2) the RTT-heterogeneity between competitive flows influences the TCP equal-share of a scheme when the bottleneck is managed by the Drop-Tail algorithm, and (3) the throughput-inversed aggressiveness and non-historical responsiveness policies should not be taken, since they cannot adapt to the change of packet loss conditions.

Chapter 3

A Fast-Converging TCP-Equivalent Window-Averaging Rate Control Scheme

3.1 Introduction

TCP is a widely-used transport protocol in the Internet. Its rate control mechanism, Additive-Increase and Multiple-Decrease (AIMD), frequently and dramatically adjusts the rate to detect the available bandwidth for transmitting packets and to avoid the packets from the sequential losses. However, TCP may be unsuitable to carry the streaming data because of its oscillatory rate. The playback of streaming data will be paused, if the sending rate of TCP is lower than the encoding rate of the streaming while no streaming data is buffered at the receiver. Although allocating a large buffer may alleviate the oscillatory rate, it prolongs the start-up latency for on-demand media traffic [GTC06] and causes an intolerable delay for interactive applications, e.g. video conference. Besides, the oscillatory rate of TCP increases the difficulty for media servers to decide which encoding rate should be taken to deliver the media for a stable user-perceived quality.

For the oscillatory-rate problem, many rate control schemes were proposed [YL00, PHP00, ROY00, JGM03, BB01, PKT99] to transmit streaming data at a smooth rate. Besides being smooth, these schemes have been suggested to be TCP-friendly [BCC98] because their controlled traffic is expected to coexist with TCP traffic in the Internet. A TCP-compatible scheme uses no more throughput than TCP, under the same conditions such as packet loss rate and round trip time (RTT). Based on the suggestion, a scheme using the same throughput as TCP is the most favorable for carrying media data because it conforms to TCP-compatibility with the highest throughput. Such a scheme is called as a TCP-equivalent scheme in this work.

TCP-equivalence is more preferred than TCP-compatibility, since the latter protects only TCP flows from starvation, but the former further ensures the fairness between TCP and the TCP-equivalent flows. In fact, current schemes are proposed for

the TCP-equivalence criterion, e.g. [YL00, PHP00, ROY00, JGM03, BB01, PKT99]. However, they pursue the criterion only under a stationary packet loss condition, i.e. the statistics of loss conditions like mean and variance are fixed at least for the whole transmission time, which however is not realistic enough. Due to the arrival and departure of burst traffic, the loss condition in the Internet may change drastically, causing these schemes having lower throughput than TCP in the long term because of their *slowly aggressive* behavior [BBF01]. Slow aggressiveness represents that these schemes cannot use the available bandwidth as immediately as TCP after the burst traffic departs. Several recent works focus on accelerating the aggressive behavior. For example, the SIMD scheme [JGM03] has the exponentially aggressive behavior. Actually, even under the stationary packet loss conditions, these schemes may still have *lower throughput* than TCP. Vojnovic and Boudec [VB05] reveal that TFRC [PFT98] has lower bandwidth than TCP if the number of inter-loss packets (i.e. packets between two loss events¹) is not periodic, i.e. its variance is not zero. In fact, the non-zero variance is the common loss conditions in the Internet [ZDP01]. Our later simulation results also show that GAIMD [YL00], IIAD [BB01], SQRT [BB01], TEAR [ROY00], and TFRCP [PKT99] cannot meet TCP-equivalence under this loss condition.

This work proposes a rate control scheme named window-average rate control (WARC). WARC is expected to better meet TCP-equivalence than the existing schemes because it takes short time to converge its rate toward TCP's whenever the available bandwidth drastically increases or decreases. Besides, when the available bandwidth keeps stationary, WARC is the first scheme providing the same bandwidth as TCP *under any distributions* of inter-loss time. Existing schemes [YL00, PHP00, ROY00, JGM03, BB01, PKT99] provide it only under some specific assumptions, e.g. the packet losses occur periodically or with a fixed probability, but these assumptions may not be realistic in the Internet. The measured result in [ZDP01] reveals that the packet loss condition experienced by an Internet flow may consist of multiple *minute-scale* stationary regions, and the time intervals of any two consecutive losses may follow an independently and identically distributed (i.i.d.) exponential distribution within a region.

The basic mechanism of rate control in WARC is called run-time estimation

¹ A loss event means an event causing a TCP flow halving its congestion window. Such an event may consist of multiple sequent packet losses. For convenience, the term "event" is ignored in this work as done in related works.

(RTE), which repeatedly estimates the mean rate of TCP and adjusts its sending rate to the estimated rate (packets/RTT). TFRCP, TFRC and TEAR also use the RTE control, but WARC differing from them estimates the mean rate by averaging the *latest fixed-number* congestion windows (CWNDs) which a TCP flow may use to control its rate. By assuming RTT is fixed², such an estimated rate represents the mean rate of the TCP flow over a *fixed* time period, instead of over a dynamic time period as that in TFRC and TEAR. The difference brings WARC use the same bandwidth as TCP under all stationary packet loss distributions, as proved later. Besides, since the number of considered CWNDs is bounded, WARC will forget the early small CWNDs and use a high increasing rate as TCP when additional bandwidth becomes available for a fixed time.

However, the RTE rate control in WARC may not be satisfying under three special but realistic conditions. First, when the available bandwidth drops abruptly, RTE may encounter many losses before reducing its rate to the available bandwidth. Second, when passing along through a link managed by the Drop-Tail queuing algorithm, RTE may exhibit a slight oscillation on rate. Third, under heavy losses, RTE would use more bandwidth than TCP. Therefore, three complements: history-reset procedure, one-RTT reduction mechanism and fluid-based timeout mechanism, are proposed for handling the three conditions, respectively.

The remainder of this paper is organized as follows. Chapter 3.2 introduces the WARC scheme, including RTE and three complements. The fairness between WARC and TCP is proved in Chapter 3.3. The smoothness and convergence rate of WARC and their tradeoffs are discussed in Chapter 3.4 according to the proofs in Appendix 2. Chapter 3.5 shows the evaluated results of WARC running in the *ns-2* simulator [NS06]. Related Work is discussed in Chapter 3.6. Finally, the applicability of WARC and conclusions are given in Chapter 3.7 and 3.8, respectively. The unfairness of TFRCP and TEAR are proved in Appendix 3.

3.2 Window-Averaging Rate Control (WARC)

3.2.1 Basic Rate-control Mechanism

The goal of a TCP-equivalent streaming rate-control scheme is to have a throughput smoother than but equivalent on average to TCP's. To achieve the goal,

² This work, like the related works [YL00, PHP00, ROY00, JGM03, BB01, PKT99], assumes that RTT is fixed in the analysis. However, in the implementation of WARC, the value of RTT refers to the smoothed RTT which is dynamically updated by the algorithm usually used in TCP.

several existing schemes such as TFRC and TEAR send packets at the mean rate of a TCP flow. Besides, these schemes repeatedly estimate the mean rate during the whole connection, since the mean rate within a period changes according to the network conditions. Such an idea to control the rate is named run-time estimation (RTE) model³ in this work.

WARC inheriting the RTE model considers two key designs. The first is how to estimate the present mean rate (packet/RTT) of a corresponding TCP flow, and the second is how often to adjust the rate of the controlled flow. For the first point, WARC gets the mean TCP rate by averaging the latest s CWNDs of a corresponding TCP flow, where s is fixed. For the second point, WARC adjusts the rate per RTT. Notably, in this work, the CWND represents the number of packets sent by a TCP flow in one RTT. Besides, packet/RTT is used as the unit of rate.

The following details the rate-control procedure between the sender and receiver of WARC. The sender sends data packets at the rate periodically assigned by the receiver. The receiver detects the packet losses with the sequence numbers of data packets, and it estimates the CWND which may be used now by a TCP flow. If the WARC receiver does not encounter losses, it will increase the CWND by one per RTT as TCP does. However, if it encounters a loss, it will reduce the CWND by a half. Next, the receiver averages the latest s CWNDs to get the new transmission rate and reports the rate to the sender, where the averaging and reporting are repeated per RTT.

According to the above description, the following formally expresses the transmission rate used in a WARC sender. Assume that the real time t is separated by RTT into multiple rounds, i.e. $t = \{1, 2, \dots, \infty\}$. Suppose that $R(t, s)$ denotes the transmission rate (packets/RTT) of the WARC sender, computed from s CWNDs and used in the t^{th} round. Then, $R(t, s)$ can be written as

$$R(t, s) = \frac{1}{\min(s, t)} \sum_{i=1}^{\min(s, t)} W(t-i), \quad (3.1)$$

where $W(t)$ is the number of packets transmitted by a TCP sender, i.e. its CWND, in the t^{th} round. A minimizing operation exists in (3.1), because there are at most t CWNDs at the initial time ($t < s$) and thus $R(t, s)$ is the average of the latest t CWNDs. When $t > s$, $R(t, s)$ will be the average of the latest s CWNDs.

When the available bandwidth suddenly increases, whether a scheme has a fast

³ We do not think that the familiar terms like "rate-based" or "equation-based" can represent such an idea. A RTE-based scheme would be a window-based scheme if using a window to control the release of packets. Besides, TEAR is not an equation-based scheme, but a RTE-based scheme as revealed later.

aggressive behavior, i.e. a high increasing rate, is determined by how fast the scheme ignores the packet loss conditions measured before the increase. WARC would have faster aggressiveness than TFRC and TEAR because WARC excludes these conditions from the rate computing after a fixed number of RTTs. Differently, TFRC and TEAR excludes these conditions after a fixed number of packet losses, say 8 losses. Unfortunately, the latter two may wait a long time to meet the first loss due to the sudden increase of bandwidth; thus they have a slow aggressive behavior. In fact, both schemes take additional rules to speed up the behavior which, however, is still conservative as shown in our simulation results later.

3.2.2 Complementary Rate-control Mechanisms

1) History-reset (HR) procedure for responsiveness under bursty-losses:

The basic control mechanism in WARC, as later analyzed in Chapter 3.4, has a conservative responsive behavior, which brings WARC a smooth rate but also a slow response to an abrupt increase of packet loss rate. Thus, a history-reset procedure is proposed to respond to the abrupt change right after a fixed number of loss events.

The procedure is invoked if the average TCP rate spanning over the latest N packet losses, denoted as $\bar{R}_{TCP}(N)$, is smaller than or equal to $1/K$ of the current rate of WARC, which can be expressed as

$$\bar{R}_{TCP}(N) \leq \frac{1}{K} R(t, s), \quad (3.2)$$

where K is a constant larger than 1, and its effect on the HR procedure would be discussed later. $\bar{R}_{TCP}(N)$ is calculated by the formula

$$\bar{R}_{TCP}(N) = \frac{3}{2} \frac{1}{N} \sum_{j=1}^N X(-j), \quad (3.3)$$

where $X(-j)$ represents the number of rounds in the *last* j^{th} epoch, i.e. the period between the j^{th} and $(j+1)^{\text{th}}$ last losses. According to [AAB05], $\bar{R}_{TCP}(N)$ got by (3.3) represents the average rate that a TCP flow may have under a loss condition where the number of inter-loss rounds is $\frac{1}{N} \sum_{j=1}^N X(-j)$. The following $\sum_{j=1}^N X(-j)$ is denoted as $S(N)$.

As Fig. 3.1 shows, if (3.2) is true at the time T , then it is implied that the packet loss condition has changed abruptly, because the fast-responsive TCP has far smaller rate than WARC. Then, the HR procedure is invoked to perform the following actions. All $W(t)$'s where $t < T - S(N)$ are eliminated from the computation of (3.1). Contrarily, the latest $S(N)$ CWNDs are retained in the rate computing. By the elimination of old

CWNDs, WARC fast jumps to the mean rate that TCP uses at the time t .

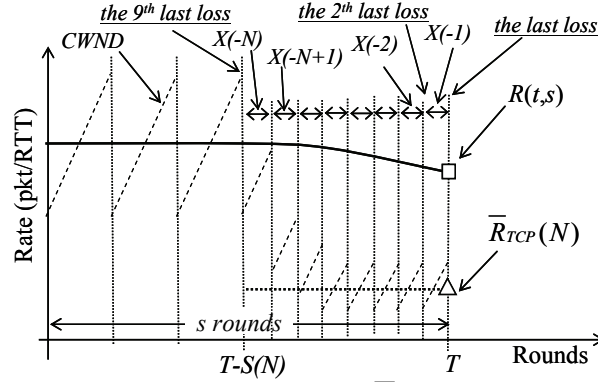


Fig. 3.1. The HR procedure would be invoked when $\bar{R}_{TCP}(N) < 1/K R(t,s)$.

Two parameters K and N in (3.2) control the threshold of invoking the history-reset procedure. K decides how many differences in rate between WARC and TCP denotes an abrupt change on packet loss condition. N decides how long the difference should persist at least before the HR procedure is invoked. With small K and N , the procedure will be invoked for small and short changes, which may damage the smoothness of WARC. Contrarily, with large K and N , the procedure may be conservatively invoked and lose its objective - fast responding to a change of packet loss condition. A tradeoff exists here obviously. We analyze the tradeoff in Chapter 3.4.4 and suggest that with $K=3$ and $N=12$, the procedure has the enough fast responsiveness and is not over-invoked to damage the smoothness.

2) One-RTT reduction procedure for smoothness under low-multiplexing networks: A flow controlled by a rate-based scheme may encounter a series of packet losses and exhibit an oscillatory behavior on the rate when passing through a low-multiplexing network [BCC98], e.g. passing alone a link managed by Drop-Tail, because the rate-based scheme does not have the per-packet ACK mechanism as the window-based scheme. A self-clocked mechanism has been proposed in [BBF01] to assist TFRC for such a case. The mechanism bounds the rate increase of TFRC according to the received rate measured at the receiver. However, the mechanism may slow down the aggressive ability of a scheme because it limits the rate increase of the scheme.

To retain the fast aggressiveness, WARC adopts the mechanism only when a packet loss occurs. When detecting a loss, the receiver of WARC will acknowledge the sender to *temporarily* reduce the transmission rate used in the next round to 7/8 of the received rate. The temporality means that the reduction holds only for one RTT,

which is long enough to prevent the flow from the sequent losses according to the simulation result shown in Chapter 3.5.3. After one RTT, the rate will return to the value computed by (3.1).

3) Fluid-based timeout (FTO) procedure for fairness under heavy-losses: A fluid-based procedure is proposed to emulate the CWNDs of TCP running under the timeout mechanism. These CWNDs will be involved in the rate computing of (3.1) to influence the rate. This complement prevents WARC from grabbing too much bandwidth when competing with TCP under heavy packet loss conditions.

WARC like TCP regards a packet loss as a timeout event if the loss occurs when $W(t) < 3$. Then, as shown in Fig. 3.2, when a timeout event is recognized, WARC enters the timeout phase from the normal phase and emulates the CWNDs by FTO instead of AIMD. WARC sets $W(t)$ to u^{-1} for I rounds where $u = \lceil \frac{RTO}{RTT} \rceil$, I is set to u , and RTO represents the retransmission timeout and is updated by the usual algorithm in TCP. Such a setting for $W(t)$ and I emulates TCP only sending one packet before RTO seconds. The variable I keeps the residual rounds that WARC stays at timeout, so it is decreased one per RTT. If the variable I is decreased to zero, WARC leaves the timeout phase. The timeout mechanism is called fluid-based because one packet sent in the RTO seconds in TCP is divided into u^{-1} packet per RTT seconds in WARC.

To emulate the back-off mechanism in the TCP timeout mechanism, another variable C is employed to record the times that packet losses occur during the phase. If another loss occurs during the phase ($C > 0$), WARC will regard the loss as the second timeout event. Then, WARC sets $W(t)$ to $(u \times 2^C)^{-1}$ and adds $u \times 2^C$ into the variable I . This prolongs the time that WARC stays in the timeout phase. Like TCP, C is limited to 6 and does not increase even when more losses occur.

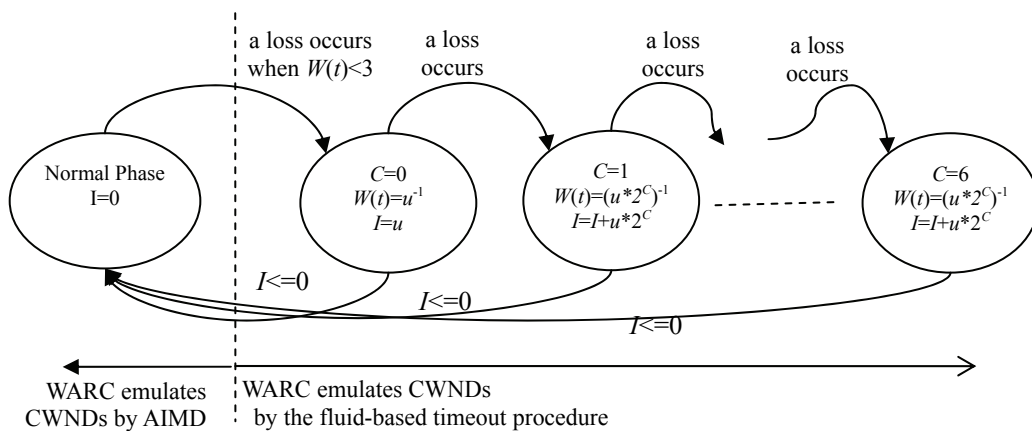


Fig. 3.2. The state diagram of the fluid-based timeout procedure in WARC

3.2.3 Pseudo Code

Fig. 3.3 presents a pseudo code to jointly describe the basic and three complementary rate-control mechanisms of WARC. The queue structures are used herein to avoid the time-consuming summing operation in (3.1). A queue Q_w keeps the latest s CWNDs and a variable S_w records the summation of all CWNDs in Q_w . Instead of repeatedly summing up the latest s CWNDs to get S_w , we simply dequeue the earliest CWND from Q_w and subtracted it from S_w , while inserting the latest CWND into Q_w and adding it into S_w , as described in the *UpdateSum* procedure. *UpdateSum* is a generic procedure to update the summation S in the queue Q with the latest value v . Similarly, we use Q_x to keep the latest N inter-loss time and S_x to store the summation of values in Q_x , to efficiently compute (3.3).

Procedure *LossHandler* is called at the receiver when a packet loss is detected. The procedure takes charge of halving the TCP CWND w , recording w in Q_w and checking whether to invoke the HR procedure by (3.3). However, if the loss occurs under the timeout phrase ($I>0$), *LossHandler* would update I , C and w , according to the rules shown in Fig. 3.2.

Procedure *UpdateW* is called per RTT. When WARC is not in the timeout phase ($I=0$), it increases w by one to emulate TCP when no packet loss is detected. However, in the timeout phase ($I>0$), *UpdateW* counts down I to update the number of remainder rounds of leaving the timeout phrase. Next, *UpdateW* sends an ACK to inform the sender the transmission rate to be used. By the One-RTT reduction procedure, the rate would be discounted by 1/8 if a loss has occurred.

<p> s : The number of CWNDs considered in the rate computing w : Current possible CWND (pkt) of a TCP connection R : Tx. rate (pkt/RTT) to be used in the next round t : The seq. num. of the current round t' : The seq. num. of round where the last loss occurred I : The number of round to leave the timeout phrase u : Ceiling value of RTO/RTT Q_w : A queue with s entries for keeping the latest CWNDs Q_x : A queue with N entries for keeping the latest inter-loss time in RTT S_w : The summation of values queued in Q_w S_x : The summation of values queued in Q_x $X \ll Y$: A math. operation equal to $X \times 2^Y$ $HasLoss$: A Boolean variable to indicate that a loss event has occurred $Ack(r)$: Send an ack to tell the sender about the transmission rate r. $HistoryReset(n)$: Keep the latest n CWNDs in the rate computing. $Enqueue(Q, v)$, $Dequeue(Q)$, $Len(Q)$, $Full(Q)$: Four queue operations to insert v into Q, get item from Q, query the length of Q and </p>
--

```

        check if  $Q$  is full, respectively.

LossHandler() // called when a loss event is detected
{
    if (I == 0) {
        if (w >= 3) {
            w = w / 2
            HasLoss = TRUE
        } else { // enter the FTO procedure
            I = u; C = 0; w = 1/u
        }
        UpdateSum(Qw, Sw, w)
        UpdateSum(Qx, Sx, t-t')
        t' = t;
        If ((Sx/Len(Qx)×1.5) × K ≤ R) // Eq. (3.3)
            HistoryReset(Sx)
    } else { // update by the FTO procedure in Fig. 3.2
        C = C + 1
        if (C > 6) then C = 6
        w = 1 / (u × (2<<C))
        I = I + (u × (2<<C))
    }
}

UpdateW() // called per RTT at the receiver
{
    if (I == 0) then w = w + 1
    else I = I - 1
    UpdateSum(Qw, Sw, w);
    R = Sw / Len(Qw)
    if (HasLoss == TRUE) then
        Ack(R×7/8) // the one-RTT reduction procedure
    else Ack(R)
    HasLoss = FALSE;
    t=t+1;
}

UpdateSum(Q, S, v) // keep S as the summation of items in Q
{
    if Full(Q) then S = S - Dequeue(Q)
    S = S + v
    Enqueue(Q, v)
}

HistoryReset(n) // keep the latest n CWNDs in Qw
{
    j = Len(Qw) - n
    while(j > 0) {
        Sw = Sw - Dequeue(Qw)
        j=j-1;
    }
}

```

Fig. 3.3 The pseudo code for the basic and three complementary rate control mechanisms of WARC

3.3 Analysis on Fairness

Herein the fairness represents that in a steady state a scheme can control a flow to have the same throughput as a TCP flow, when both flows experience the same

stationary packet loss condition. This section will prove the fairness of WARC. Among another three RTE-based schemes, TFRC has been shown in [VB05] that it meets the fairness only if an additional assumption is given, that is, the inter-loss time is periodic, i.e. constant. Besides, we proved in Appendix 2 that TEAR uses lower throughput while TFRCP may use higher throughput than TCP when the inter-loss time is not periodic. The fairness analyses of other schemes aim to demonstrate that the difference between WARC and other RTE-based schemes does cause the different abilities in meeting the fairness. The effect of the TCP timeout mechanism is ignored in the analysis below.

Recall that (3.1) shows the rate used by a WARC flow in the round t . Suppose that when the time $t=t_0$ the WARC flow converged its rate on a steady-state rate. Then, for $t>t_0$, the long-term average throughput of the flow, $E[T_{WARC}]$, can be expressed as

$$E[T_{WARC}] = \lim_{k \rightarrow \infty} \frac{\sum_{t=t_0}^{t_0+k} R(t, s)}{k}. \quad (3.4)$$

Substituting (3.1) into (3.4) yields

$$\begin{aligned} E[T_{WARC}] &= \lim_{k \rightarrow \infty} \frac{\sum_{t=t_0}^{t_0+k} \left(\frac{1}{s} \sum_{i=1}^s W(t-i) \right)}{k} \\ &= \frac{1}{s} \lim_{k \rightarrow \infty} \frac{\sum_{i=1}^s \sum_{t=t_0}^{t_0+k} W(t-i)}{k} \\ &= \frac{1}{s} \sum_{i=1}^s \left(\lim_{k \rightarrow \infty} \frac{\sum_{t=t_0}^{t_0+k} W(t-i)}{k} \right). \end{aligned} \quad (3.5)$$

Since $W(t)$ represents the window size of a TCP flow in the round t , (3.5) can be rewritten as

$$E[T_{WARC}] = \frac{1}{s} \sum_{i=1}^s E[T_{TCP}] = E[T_{TCP}]. \quad (3.6)$$

Equation (3.6) shows that WARC meets the fairness.

3.4 Analyses on Smoothness, Aggressiveness and Responsiveness

A tradeoff exists within three properties: smoothness, aggressiveness, and responsiveness. Performing a smooth rate may imply the existence of slow aggressiveness and responsiveness, i.e. taking longer time to converge toward the new steady-state rate. The section first discusses the analytical results of WARC in terms

of smoothness, aggressiveness, and responsiveness. The analysis procedures are described in Appendix 2. Secondly, we discuss the tradeoff within the three terms in WARC, and compare it with that in SIMD, GAIMD, and IIAD. For the schemes without the analyses, e.g. TFRC, TFRCP and TEAR, we compare WARC with them by the simulation later.

3.4.1 Smoothness

The smoothness [JGM03] is defined as the average coefficient-of-variation (CV) of the rate (packets/RTT) between two consecutive packet losses, as

$$\overline{CV}[w] = \frac{\sqrt{E[Var[w]]}}{E[w]} = \frac{\sqrt{E[Var[w]]}}{W}, \quad (7)$$

where w is the sending rate, i.e. the number of packets sent in one RTT, and W denotes $E[w]$. Because the rate w of a TCP-equivalent flow is controlled based on the loss conditions, the $E[w]$ of the flow can represent the degree of loss conditions experienced by the flow. Thus, by giving a specific $E[w]$, (7) represents the smoothness of a scheme performing under a specific degree of loss conditions. According to the analysis in Appendix 2.1, Fig. 3.4 plots the $\overline{CV}[w]$ of WARC, normalized with that of TCP [JGM03], over various $E[w]$'s in order to show the smoothness effect provided by WARC under different degrees of loss conditions. The normalized $\overline{CV}[w]$'s of SIMD and GAIMD, derived in [JGM03], are also plotted for comparison. As shown in Fig. 3.4, the curves of WARC(120), WARC(160), WARC(240), and WARC(320) cross over that of GAIMD respectively at $W=23.625$, 31.5, 47.25, and 63. That is, WARC has a smoother rate than GAIMD(1/8) when its parameter s is configured to a value larger than $120/23.625$ (i.e. 5.07) times of W . According to Fig. 3.4 of [JID04], since the maximum CWND of most TCP flows are smaller than 31.62, we set the parameter s to $5.07*31.62$ (i.e. 160) in the following work. Notably, WARC does have a smoother rate than TCP, i.e. the normalized $\overline{CV}[w]$ is smaller than 1, although the four curves of WARC plotted in Fig. 3.4 seem to increase linearly. Their slopes become smaller than 1 at $W=150\sim 200$ and their $\overline{CV}[w]$'s are smaller than TCP's one even when W is infinite.

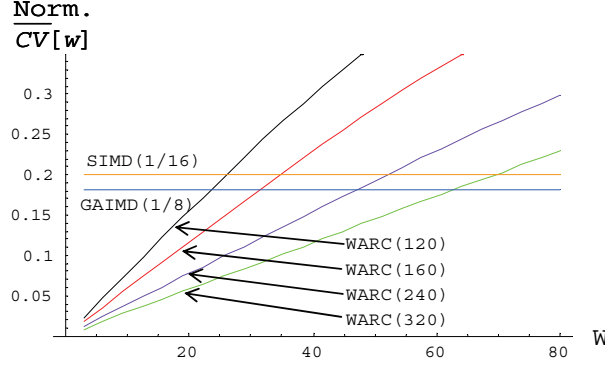


Fig. 3.4. The smoothness effects of WARC, GAIMD, and SIMD, relative to that of TCP.

3.4.2 Aggressiveness

A scheme is said as aggressiveness if it takes short time to increase its rate in response to a step increase of available bandwidth [RKL01]. By such a concept, Shudong *et al.* in [JGM03] defined⁴ the index $1/Aggr(m)$ as the time required by a scheme to increase its rate by a factor of m . This work follows [JGM03] showing the analysis of WARC by $1/Aggr(m)$ to easily compare WARC with GAIMD, SIMD, and IIAD, which were analyzed in [JGM03]. According to the analysis in Appendix 2.2, the aggressiveness of WARC is written as

$$\frac{1}{Aggr(m)} = \begin{cases} \frac{1}{3}(W + \sqrt{W^2 + 18msW - 18sW}) & (m \leq \frac{2}{3}W + \frac{s}{2W}) \\ (m - \frac{2}{3})W + \frac{1}{2}s & (m > \frac{2}{3}W + \frac{s}{2W}). \end{cases} \quad (3.8)$$

Fig. 3.5(a) compares $1/Aggr(m)$ of WARC(160) with that of GAIMD(1/5,1/8), SIMD(1/16) and IIAD(1,2/3) in different m . WARC like SIMD takes shorter time to converge than GAIMD and IIAD, which is particularly obvious when m is larger than 1.6. Besides, Fig. 3.5(b) shows the tradeoff between aggressiveness and smoothness for the four TCP-friendly schemes when $m=4$. Obviously, WARC has better tradeoff than GAIMD and IIAD. For example, when the four schemes have $\overline{CV}[w]=0.04$, WARC takes merely 150 RTTs to converge the rate, but GAIMD and IIAD take 445 and 1080 RTTs, respectively.

⁴ Actually, Shudong *et al.* in [JGM03] first defined the index $Aggr(m)$, but they plotted and discussed all results by $1/Aggr(m)$. $Aggr(m)$ can be considered as the average acceleration used by a scheme to increase its rate by a factor of m . Assume that the increased amount of the rate is always one for any m . Then, the average acceleration is the inverse of the time required by the scheme to increase its rate, i.e. $1/Aggr(m)$.

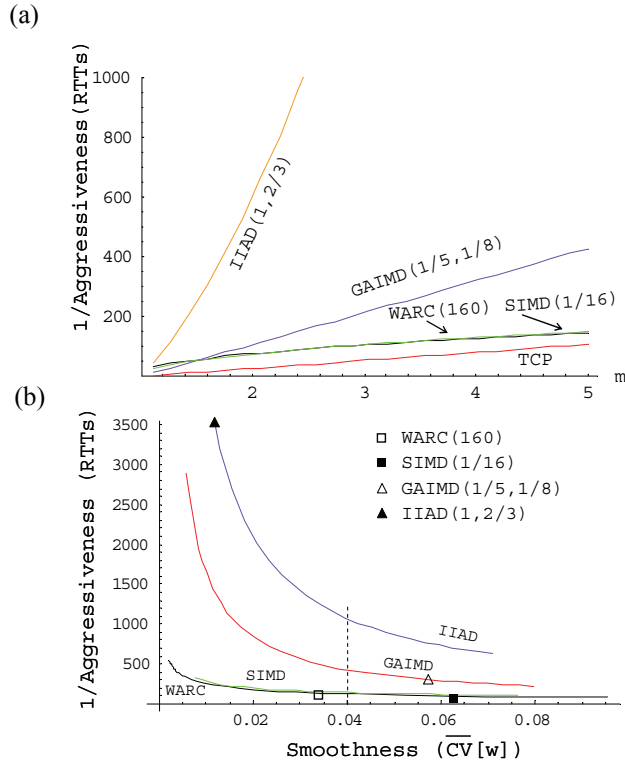


Fig. 3.5. (a) The aggressiveness indices of WARC, SIMD, GAIMD and IIAD under different increasing factor m 's. The initial average window size before the bandwidth change is 20. (b) The tradeoff between aggressiveness and smoothness when $m=4$.

3.4.3 Responsiveness

The responsiveness [JGM03] is defined as the inverse of the number of packet losses encountered by a scheme before the scheme decreases its rate with a factor of m . A scheme with a high responsiveness means that it decreases its rate in a large step per packet loss on the average. Similar to $1/Aggr(m)$, $1/Resp(m)$ is used in the following discussion. According to Appendix 2.3, the responsiveness of WARC can be expressed as

$$1/Resp(m) = \begin{cases} \frac{3 \cdot s \cdot m}{2W} + \frac{4}{3} & \text{if } m < b(K) \text{ (by the RTE control)} \\ N & \text{if } m \geq b(K) \text{ (by the HR procedure),} \end{cases} \quad (3.9)$$

where $b(K)$ represents a separation whose value is displayed as

$$b(K) \approx 1.1 \times K.$$

For the situation where m is lower than the separation, the responsiveness of WARC is ruled by the RTE control. Otherwise, it is ruled by the HR procedure.

$1/Resp(m)$ represents how many packet losses a scheme would encounter before converging its rate to $1/m$ of the original one. As shown in (3.9), WARC has two types of responsive behaviors individually contributed by the RTE control and the

HR procedure. To display the two behaviors, Fig. 3.6(a) plots the $1/Resp(m)$ of WARC over various m . When m is small, WARC runs under the RTE control and the number of the encountered losses before its convergence is direct proportional to m . Also, WARC with large s would take more losses since it is supposed to have a smoother rate. However, when m is large, the HR procedure would be invoked, leading that WARC converges its rate right after a fixed number of losses. Fig. 3.6(b) further displays the different effects on the responsiveness brought by various configurations.

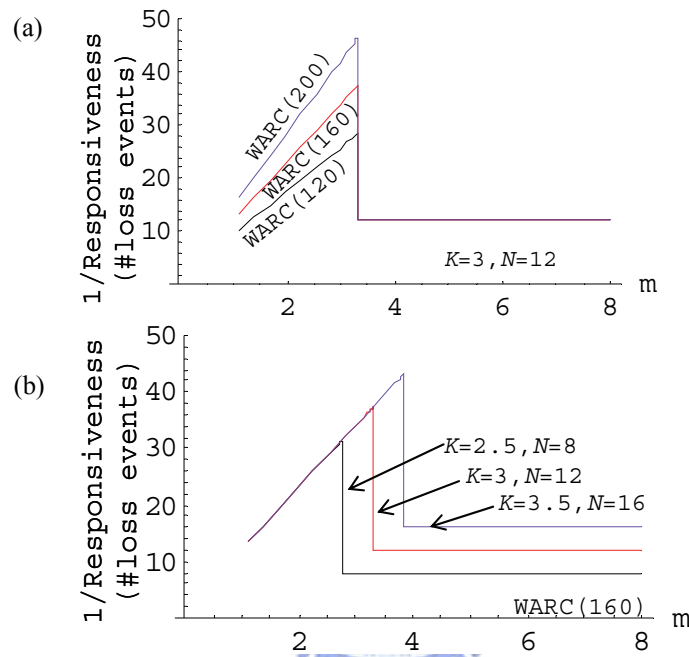


Fig. 3.6. The responsiveness of WARC under varied decreasing factor m 's. The initial average window W before bandwidth change is 20. (a) Different s . (b) Different pairs of K and N .

Fig. 3.7(a) compares the responsiveness of WARC(160) with that of other schemes. Although WARC takes more packet losses to converge its rate when the change of loss condition is small, it takes merely half of losses as SIMD and IIAD to converge its rate when the change is large. Unlike other schemes, WARC takes different responsive behaviors for different degrees of loss change, which makes WARC *keep its rate smooth under small and temporary changes of loss condition, but quickly respond to the abrupt change right after N losses*. Fig. 3.7(b) shows the tradeoff between smoothness and responsiveness for the four schemes. As expected, WARC will take more packet losses before it converges its rate if the loss change is small as shown in the top of the figure, but it will take fewer losses if the change is large as shown in the bottom.

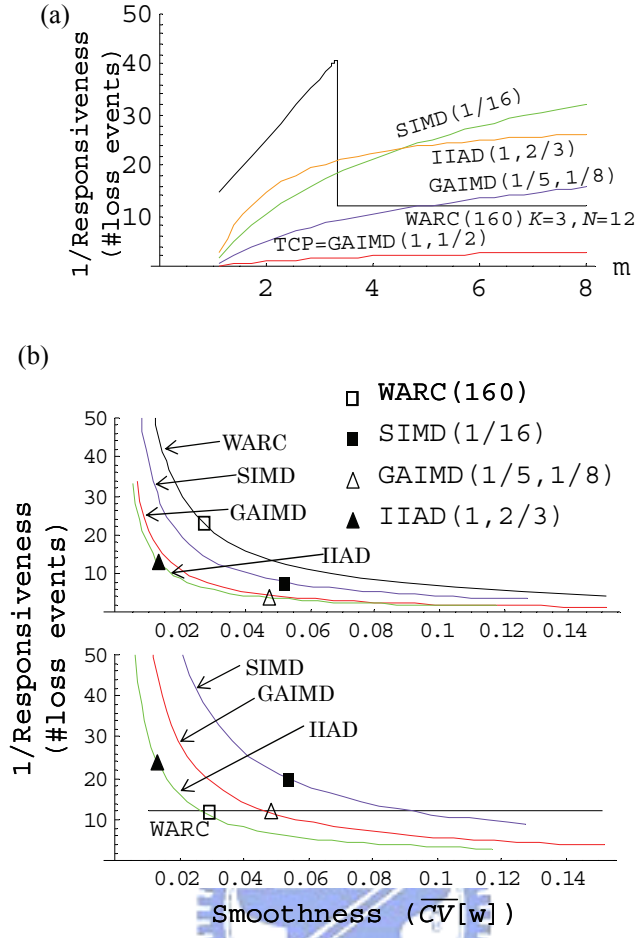


Fig. 3.7. The responsiveness of WARC, SIMD, GAIMD and IIAD.
 (a) Responsiveness under different increasing factors.
 (b) Responsiveness versus smoothness when $m=1.7$ and $m=4$

3.4.4 False Positive of the HR procedure

Although the HR procedure provides WARC fast responsiveness when the available bandwidth reduces dramatically, the smoothness of WARC may be degraded if the procedure is invoked unnecessarily, i.e. invoked for a temporary and small reduction. We call such an unnecessary invocation as a false-positive case. The following analyzes the probability on the false-positive invocation when the HR procedure is given a specific configuration of (K, N) . The analysis will suggest that $(K, N)=(3, 12)$ should be a suitable configuration.

Herein we define a false-positive invocation as that the HR procedure is invoked when the inter-loss time is stationary. Assume that in the steady state $R(t, s)$ equals to the mean throughput of a TCP flow, which approximates $1.5 E[X]$ where $E[X]$ denotes the mean inter-loss time in RTT [AAB05]. By the definition, under the stationary inter-loss time, the probability that (3.2) is true, i.e. HR is invoked, can be expressed as

$$\begin{aligned}
P[\bar{R}_{TCP}(N) \leq \frac{1}{K} R(t, s)] & \\
&= P\left[\frac{3}{2N} \sum_{j=1}^N X(-j) \leq \frac{1}{K} \frac{3}{2} E[X]\right] \\
&= P\left[\frac{1}{N} \sum_{j=1}^N X(-j) \leq \frac{1}{K} E[X]\right].
\end{aligned} \tag{10}$$

Obviously, the probability is controlled by two parameters: K and N . According to the Internet loss conditions reported in [ZDP01], we assume that the inter-loss time X experienced by a flow follows an i.i.d. exponential distribution with a parameter $\lambda = 1/E[X]$. Then, since the term $\sum_{j=1}^N X(-j)$ in (3.10) represents the sum of N exponential random variables, it forms a gamma distribution (N, λ) by the probability theory. Therefore, (3.10) could be rewritten as

$$\begin{aligned}
&P\left(\bar{R}_{TCP}(N) \leq \frac{1}{K} R(t, s)\right) \\
&= P\left(\sum_{j=1}^N X(-j) \leq \frac{N}{K} E[X]\right) = P\left(\sum_{j=1}^N X(-j) \leq \frac{N}{K\lambda}\right) \\
&= F_{\text{gamma}(N, \lambda)}\left(\frac{N}{K\lambda}\right) = F_{\text{gamma}(N, 1)}\left(\frac{N}{K}\right),
\end{aligned} \tag{11}$$

where $F_{\text{gamma}(a, b)}$ is the cumulative distribution function of the gamma distribution with the shape parameter a and the scale parameter b . λ is set to 1 since its value does not affect the probability.

According to (3.11), Fig. 3.8 plots the numerical results of the probability of false-positive invocation over different K and N . Assume we expect that WARC has 10^{-3} of the false positive probability; then, the possible configuration of (K, N) may include (2.5,16), (3,12) and (3.4,10). The probability of 10^{-3} implies that the HR procedure may be unnecessary invoked once per 1000 losses. The probability should be low enough. Assume W is the mean CWND of a flow and the inter-loss time in RTT is $W/1.5$ [AAB05]. Also, the possible range on W is from 6 to 30 [JID04] and the possible range on RTT is from 0.050s to 0.300s. Then, the mean time spanning 1000 losses in second is

$$\frac{1}{0.3-0.05} \int_{0.05}^{0.3} \left(\frac{1}{30-6} \int_6^{30} \left(1000 \times \frac{W}{1.5} \times RTT \right) dW \right) dRTT = 2100,$$

which should be long enough because the period of a stationary state in the Internet mostly is shorter than 600 seconds [ZDP01].

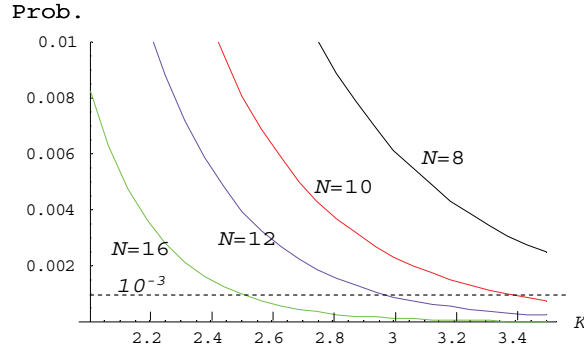


Fig. 3.8. The probability of the false-positive invocation of the HR procedure

3.5 Simulation Results

This section evaluates and shows that WARC has better behaviors than eight TCP-friendly schemes in terms of 4 properties: fairness, smoothness, aggressiveness and responsiveness, by running the *ns-2* simulator [NS06]. Herein a TCP-friendly scheme represents it aims to be TCP-equivalent. Table 3.1 describes the control parameters of each scheme used in the simulation, which were suggested individually according to the corresponding papers.

TABLE 3.1. THE CONTROL PARAMETERS USED IN EACH SCHEME

SCHEME	CONTROL PARAMETERS	REF.
WARC	$s=160, K=3, N=12$	
GAIMD	$\alpha=0.2, \beta=0.125$	[YL00]
IIAD	$\alpha=1.0, \beta=0.67, k=1, l=0$	[BB01]
SQRT	$\alpha=1.0, \beta=0.67, k=0.5, l=0.5$	[BB01]
SIMD	$\beta=0.0625, k=-0.5, l=1$	[JGM03]
AIAD/H	$\beta=0.25, k=0, l=0$	[JGM03]
TFRCP	Interval=5 seconds	[PKT99]
TFRC	The number of samples=8	[FHP00]
TEAR	The number of samples=8	[ROY00]

3.5.1 Fairness

3.5.1.1 Simulation topologies

WARC and other selected schemes are evaluated in three cases. The former two use an artificial packet loss link, as plotted in Fig. 3.9, to drop packets based on mathematical model. Such a link is usually for testing the fairness between TCP and the TCP-friendly scheme because it ensures the identical loss conditions experienced by any two passing flows, which is a basic assumption for these schemes to perform fairness. The mathematical model used here follows a general exponential distribution,

which has two degrees of freedom and thus allows its coefficient-of-variation to be changed while fixing its mean, or vice versa. By the general exponential distribution, we can test the fairness behavior in term of different means and different CVs, individually. Sufficient bandwidth is allocated for this link to prevent the packets being dropped because of overflow.

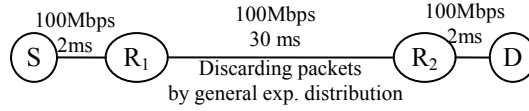


Fig. 3.9. The artificial loss-link topology is used to provide the same loss condition for any two flows running through the link R_1 - R_2 .

The third case makes all flows compete for a single bottleneck, which may be managed by the Drop-Tail or RED algorithms. The test is used to verify whether a TCP-friendly scheme is safe to deploy in the Internet. Fig. 3.10 shows the dumbbell topology used in this test. The v TCP-friendly flows compete with v TCP flows for a single bottlenecked link. All flows have backlogged data for the whole testing period. Such a scenario tests whether v TCP-friendly flows can share the same bandwidth with v TCP flows under different levels of congestion. The capacity of the congested link is configured to 15Mbps or 60Mbps, and that of other links to 100Mbps. The value v is changed from 1 to 64. The queue size is 1.5 and 2.0 of bandwidth-delay product when R_1 is managed by Drop-Tail and RED, respectively. The propagation delay of the links from the sources to R_1 or from R_2 to the destinations is uniformly distributed between 10 to 30 ms.

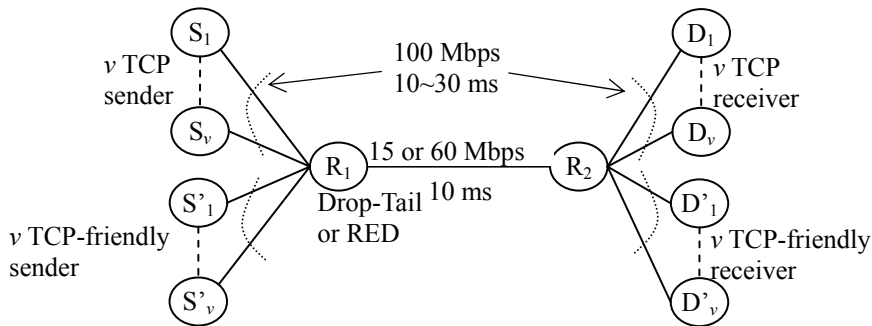


Fig. 3.10. The dumbbell topology used to test the fair sharing between TCP and TCP-friendly

3.5.1.2 Case I - Under artificial-loss links with different means

Fig. 3.11 displays the normalized throughput of each TCP-friendly scheme, compared with TCP, over the links with different means of the number of inter-loss packets. The link discards packets every time when receiving P packets, where P is a

general exponential distributed random variable with a small coefficient-of-variation ($CV[P]=0.01$). The results are averaged from three 2200-seconds runs and the data in the beginning 200 seconds is ignored. Error bounds of the results are not presented because the three runs almost have the same results.

Fig. 3.11 shows WARC like SQRT has better fairness than other schemes under various loss ratios. Particularly under the cases where the ratio >0.03 , WARC provides 0.8~1.1 of TCP's throughput while GAIMD, SIMD⁵ and IIAD provide lower throughput than TCP's (0.4~0.6). The curve "WARC w/o TO" in the right part of Fig. 3.11 represents the result of WARC without the fluid-based timeout mechanism. By comparing this curve with that of WARC, it is demonstrated that the timeout mechanism proposed in Chapter 3.2.2.3 does prevent WARC from using more bandwidth than TCP under the heavy-loss conditions. Moreover, although GAIMD, SIMD and IIAD use the TCP timeout mechanism to control their rate under heavy-losses, their lower throughput reveals that directly using the TCP timeout mechanism does not ensure fairness, which results from that they trigger the timeout mechanism more frequently than TCP, according to our further observation.

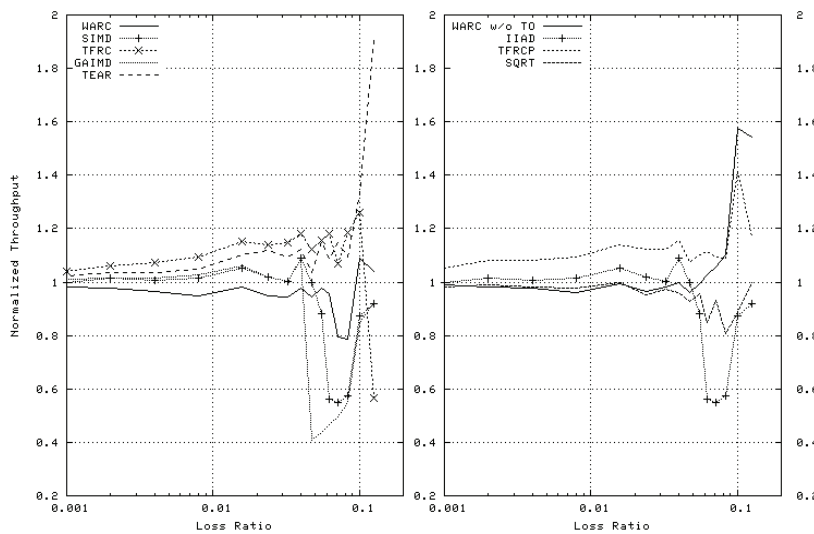


Fig. 3.11. The throughputs of TCP-friendly schemes normalized with that of TCP under different loss probabilities. Results are separately plotted in two parts for clarity.

3.5.1.3 Case II - Under artificial-loss links with different CVs

The following reveals the fairness behaviors of schemes under the link with different coefficient-of-variations of inter-loss time. The used link drops packets per T

⁵ The congestion control parameters of SIMD used in Case I is specially calculated for $CV[P]=0$ to match the loss model used here. Under this case, the original parameters [JGM03], optimized for $CV[P]=1$, will cause SIMD to get lower bandwidth than its result now plotted in Fig. 10.

seconds where T follows a general exponential distribution with $E[T]=5$ and $CV[T]$ uniformly increases from 0 to 1. In this testing case, the link drops packets based on the escaping time from the last loss, instead of the number of received packets as that in Case I, since dropping-by-time would be more realistic to emulate the loss conditions in the highly multiplexing network like the Internet [BCC98]. Actually, we also observed the fairness behaviors of schemes under the link with different coefficient-of-variations of inter-loss packets. However, the result is skipped because it is similar to that in Fig. 3.12.

Fig. 3.12 shows that WARC uses the same throughput as TCP under all $CV[T]$'s because it is supposed to perform fairness under stationary losses, as proved in Chapter 3.3. Contrarily, most schemes only have the fairness as $CV[T]=0$, i.e. as the loss occurs periodically, because the assumption of periodic losses is given in these schemes. Actually, this assumption does not consist with the loss pattern in the Internet. The inter-loss time in Internet may approximate an i.i.d. exponential distribution [ZDP01], which is the link with $CV[T]=1$. Under $CV[T]=1$, WARC provides the fairness, but GAIMD and TFRC only have about 80% throughput of TCP while TEAR, IIAD, SQRT, and AIAD/H have 60% on the average.

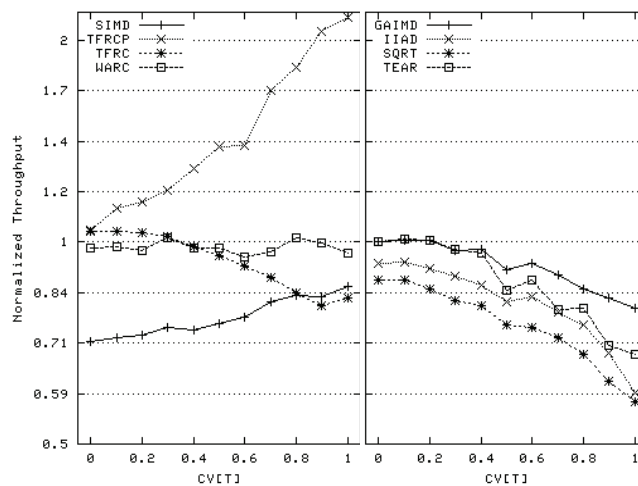


Fig. 3.12. The throughputs of the TCP-friendly schemes normalized with that of TCP under the artificial-loss links with different $CV[T]$. Results are separately plotted in two parts for clarity.

The TFRCP and SIMD flows exhibit different trends from others. The distinctness of TFRCP is due to the convex TCP throughput equation [VB05] while that of SIMD results from that SIMD computes the congestion control parameters under the loss model with $CV[P] \sim 1$ [JGM03]. We also plot the curve SIMD/Period for the SIMD with the parameters computed under $CV[P]=0$, which does not use the same

throughput as TCP.

3.5.1.4 Case III - Under Drop-Tail or RED

Fig. 3.13 and 3.14 show the results that TCP individually competes with five TCP-friendly schemes under the four configurations of the congested link: (Drop-Tail, 15Mbps), (Drop-Tail, 60Mbps), (RED, 15Mbps), and (RED, 60Mbps). For each configuration, five schemes are separately shown in two figures for clearness. As shown in Fig. 3.13(a)(c) and Fig. 3.14(a)(c), WARC almost has the similar behavior as TFRC to equally share the bottleneck bandwidth with TCP under the four configurations. As shown in Fig. 3.13(c), WARC, as well as TFRC and TEAR, has slightly lower throughput than TCP, because the rate-based control mechanism taken in the three schemes may experience a bit higher loss ratio than the window-based control mechanism taken in GAIMD and SIMD. These additional losses results from that the former does not really control the data packet transmission by the received acknowledgement packet and thus cannot respond to the overflow of the Drop-Tail queue within a RTT.

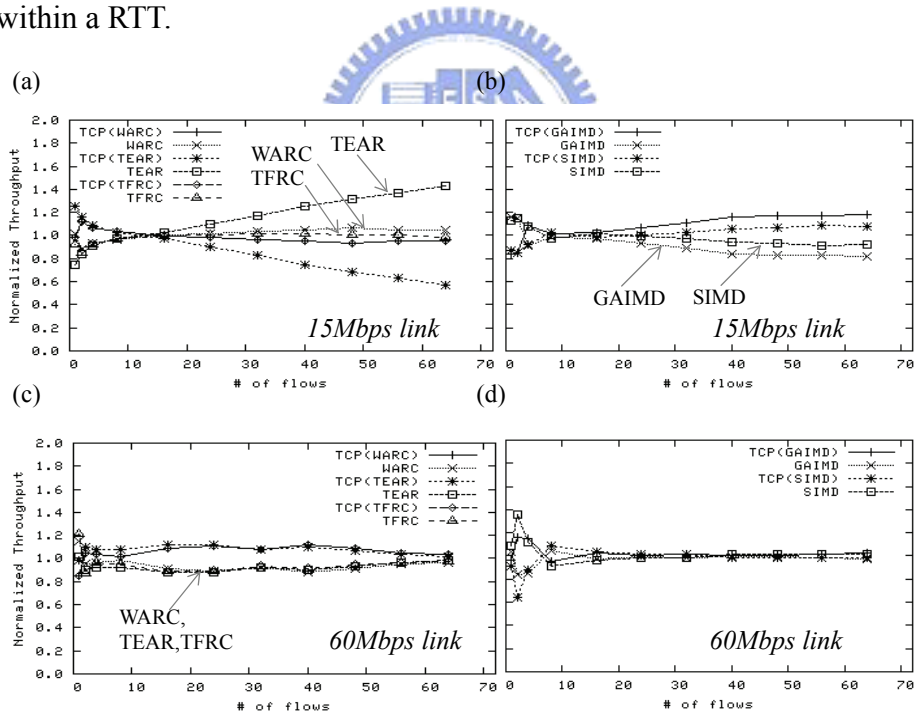


Fig. 3.13. The competing results between TCP and five TCP-friendly schemes under the links managed by Drop-Tail are shown. TCP(*X*) plots the average normalized throughput of TCP flows which compete with the flows controlled by the scheme *X*.

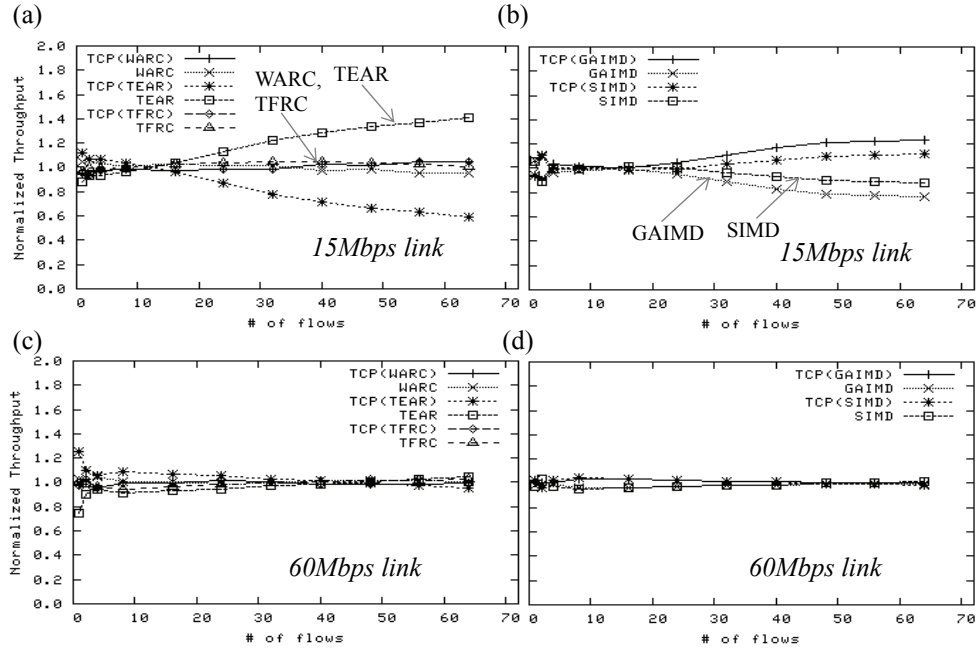


Fig. 3.14. The competing results between TCP and five schemes under the links managed by RED are shown.

Fig. 3.13(a)(b) and Fig. 3.14(a)(b) display that TEAR and SIMD may have unequal sharing to TCP when the number of flows ν exceeds 16. Our further study finds that under such conditions, TCP may encounter many losses, so controlling its rate with the timeout mechanism. In such a situation, as mentioned in Chapter 5.1.2, TEAR may use more bandwidth while SIMD and GAIMD may use less one than TCP.

3.5.2 Smoothness

Herein the smoothness degree of a WARC flow is revealed and compared with that of the flows carried by TCP and other TCP-friendly schemes. The smoothness degree is observed over different time scales, because a scheme would be more favorable to control the rate of a streaming flow if it can provide a smooth rate *even under a small time-scale*. We define the smoothness metric as follows. The rate of a flow, R , is sampled per 0.1 second. The $CV_k[R]$ is the coefficient-of-variation of R ($CV[R]$) calculated over k samplings and represents the smoothness of a flow at the time scale k .

The topology shown in Fig. 3.10 is applied for the testing. Ten TCP flows compete with ten WARC flows for a 40 Mbps-link. RED is employed as the queue management policy in this link and the queue size is set as twice bandwidth-delay product. The competition continues 2200 seconds and the data in the former 200

seconds is eliminated from the statistics of results. We also run such a competition for other TCP-friendly schemes.

For each TCP-friendly scheme, Fig. 3.15 plots its $CV_k[R]$ normalized with that of TCP over different time scale k 's. WARC, as well as TFRC and TEAR, has better smoothness than other schemes because of its RTE control. It is also demonstrates that the smoothness in WARC does not be destroyed by its fast aggressive and responsive capabilities. Moreover, the results in this figure shows these TCP-friendly schemes do provide smoother rate than TCP in the long term, observed from the normalized $CV_{512}[R] \sim 0.5$. Lastly, because these schemes avoid largely changing their rates between two losses, they have better smoothness at the time scale 16 (1.6 second) which approximates the average inter-loss time in the testing.

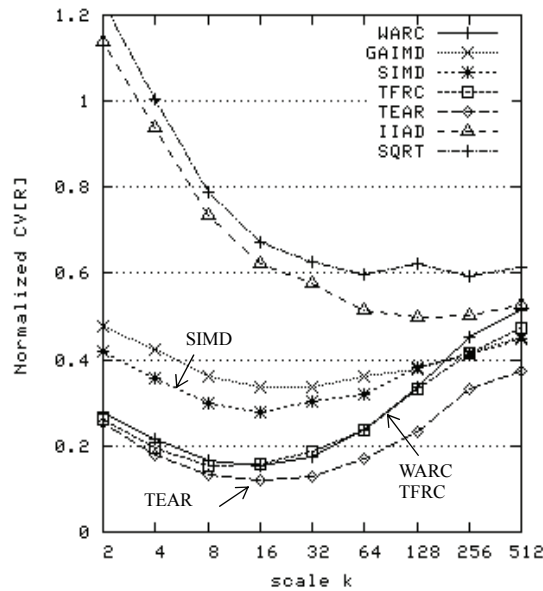


Fig. 3.15. The smoothness of each scheme over different time scales.

3.5.3. Aggressiveness and Responsiveness

To demonstrate the fast-convergent behavior in WARC, an on/off CBR traffic with obviously different rates between on and off periods is used, as shown in Fig. 3.16. Such traffic brings dramatic changes on the packet loss condition and thus provides the required transient-state scenarios. In [BBF01], such an oscillating CBR traffic is used to observe whether GAIMD, TFRC, IIAD, and SQRT use the same bandwidth as TCP. The bottleneck in the test is a 16Mbps link managed with Drop-Tail, where the rate of the on/off CBR traffic oscillated between two values, 15Mbps and 10Mbps, to vary the bandwidth available for the TCP-friendly flow to 1Mbps and 6Mbps, respectively. The propagation delay of flows was 30 ms, and the

queue size was set to 1.5 times the bandwidth-delay product.

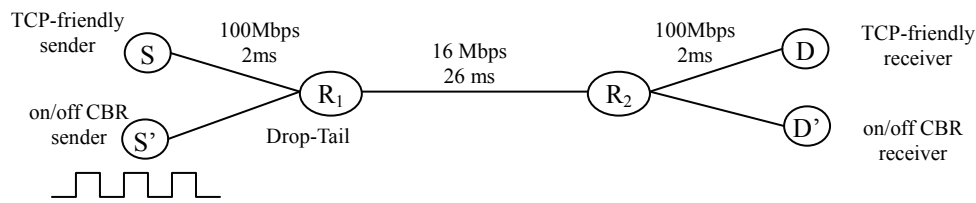


Fig. 3.16. The topology with oscillating CBR background traffic, used to test TCP-friendly schemes in terms of aggressiveness and responsiveness

Fig. 3.17 first shows that WARC has the fast aggressiveness when additional bandwidth becomes available at the 450th second. WARC like SIMD converges toward the new rate within about 20 seconds, which is shorter than GAIMD, TFRC and TEAR. The fast aggressiveness results from that WARC forgets the measured packet loss conditions earlier than a fixed number of RTTs, but TEAR and TFRC cannot do it, as mentioned in Chapter 3.2.1.

On the other hand, by using the HR procedure, WARC like TFRC and TEAR has the fast responsiveness at the 600th second. A fine look at Fig. 3.17 around the 600th second reveals that the HR procedure in WARC is invoked at 603th second. Once the HR procedure is invoked, the rate of WARC immediately reduces to the expected TCP rate. Fig. 3.18 shows the number of losses encountered by each schemes between the 600th and 620th second, normalized with that by TCP. Obviously, by using HR, WARC encounters fewer losses than GAIMD and SIMD before reducing its rate toward TCP's.

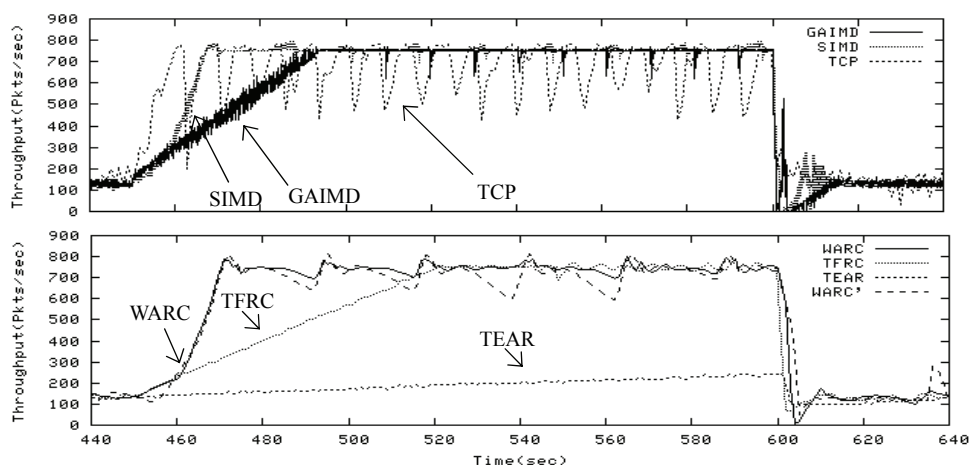


Fig. 3.17. The comparison between five TCP-friendly schemes on aggressiveness and responsiveness under the on/off CBR background traffic

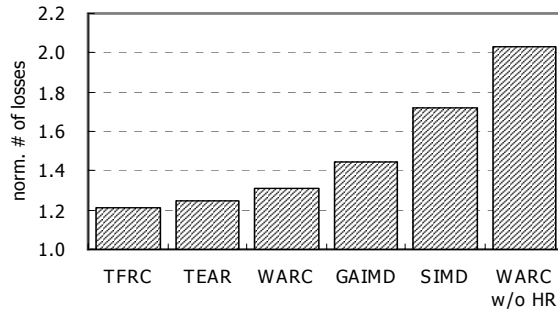


Fig. 3.18. The number of losses encountered by WARC, WARC without HR, and other four schemes between the 600th ~ 620th second are plotted, which are normalized with that by TCP.

Notably, in Fig. 3.17 the curve WARC' plots the rate of a flow controlled by WARC without the one-RTT reduction procedure. Obviously, the curve is more oscillatory than that of WARC since the flow encounters more packet losses when the queue of the bottleneck is overflowed. The result confirms that the one-RTT reduction procedure prevents a WARC flow from the sequential losses and the rate oscillation when the flow alone passes through a Drop-Tail link.

Next, we observe whether WARC uses the same bandwidth as TCP under links with various frequencies of oscillating CBR traffic. In this test, the length of the on/off period of the CBR traffic is varied from 2 seconds to 128 seconds. Fig. 3.19 plots the average rates of WARC and other TCP-friendly schemes, normalized with that of TCP for comparison. Obviously, when the oscillating period is smaller than 16, all schemes intend to keep its rate smooth, so they get less bandwidth than TCP. However, under the link with long oscillating period, since WARC like SIMD has the fast aggressiveness to immediately use the available bandwidth, it can use the similar bandwidth as TCP.

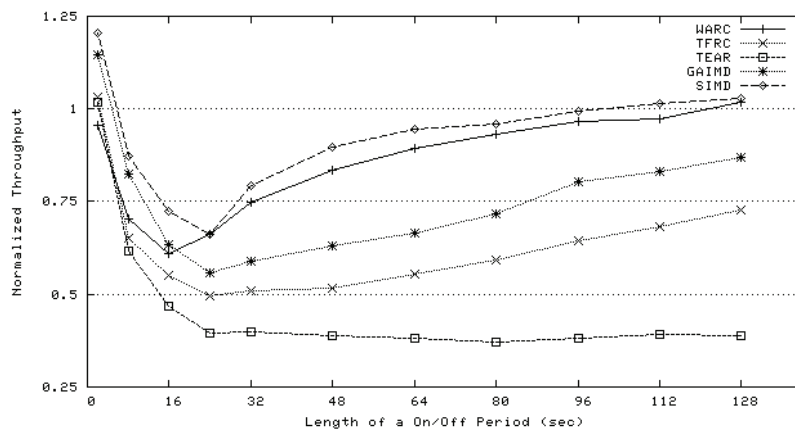


Fig. 3.19. The normalized throughputs of TCP-friendly schemes under an oscillating CBR traffic

3.6 Related Work

This work focuses on congestion control schemes which use smooth and TCP-equivalent rate to send packets over the Internet. These schemes like TCP NewReno and SACK consider packet loss as congestion signals to adjust their rates. Actually, besides packet losses, the RTT variation can be used to detect congestions, as in TCP Vegas [BP95] and FAST [JLH07]. Although RTT-based schemes provide a smooth rate too, they may use the network bandwidth unequal to that used by loss-based versions of TCP [TWH05], which still take charge of carrying most traffic over the Internet. C. Zheng and V. Tsoussidis [ZT06] recently proposed a scheme using both losses and RTTs, which may be a solution to the unfairness problem.

All the schemes mentioned above have a low barrier to deploy into the Internet, because they detect congestion by the packet loss or/and delay and need not any feedback from routers. However, adjusting rate only by loss and delay may not work well in high Bandwidth-Delay Product (BDP) networks [LPW03][KHR02], which a future Internet may belong to. Therefore, researches, e.g. [KHR02] and [XSS05], consider a tight cooperation model between congestion control schemes and routers. XCP [KHR02] is such a control scheme and requires multiple-bits congestion-related feedback from routers. However, since there is no space in IP header to carry these bits, XCP has a high barrier to be deployed in the Internet. Recently, VCP is proposed in [XSS05] to use two bits to provide the similar effect as XCP. Although VCP overcomes the problem in XCP, it still requires that all routers on the path encode the level of congestion in the IP header. Therefore, the schemes which require feedback from routers, e.g. XCP or VCP, may not be soon applied in the Internet to carry streaming traffic.

Besides the congestion control, other factors must also be considered when designing a protocol for carrying streaming traffic. E. Kohler *et al.* [KHF06] discussed these factors in depth, and proposed the Datagram Congestion Control Protocol (DCCP). DCCP allows free selection of a congestion control scheme, and therefore is the most realistic means for practical use of schemes addressed in this study. The protocol currently only includes two schemes, namely TCP-like and TFRC. We strongly encourage adding other schemes to this protocol.

3.7 Applicability

There are two possible applicable positions in end hosts to deploy WARC. The first is in the network transport layer, which is a heuristic position because WARC is proposed to play a role like AIMD, which is in a transport-layer protocol, i.e. TCP. However, instead of modifying TCP to support WARC, we suggest adding WARC into DCCP [KHF06], a transport protocol recently designed for streaming. Compared to TCP tightly binding AIMD, DCCP leaves options for the rate control schemes, and thus WARC can easily stand as one of the rate-control options in DCCP. Besides, DCCP removes several mechanisms in TCP, e.g. fast retransmission and fast recovery, which are unnecessary for streaming and may cause additional packet delay.

In fact, deployment in the transport layer is also favorable to promote WARC for existing streaming applications. Today most applications access the Internet through a socket interface, which wraps the complex network protocols like TCP/IP in a standard I/O functions, e.g. open/close and read/write. Therefore, there is no barrier to enable these applications using new rate controls or transport protocols because the socket interface is unchanged. However, the con of this deployment is the necessary of kernel programming of operating system (OS), since the transport layer is usually implemented inside the kernel. Therefore, it is difficult to implement WARC into the kernel if the OS does not open the programming interface for the deployment. Fortunately, it is difficult only but not impossible. Today, even the commercial OSs, like Microsoft Windows, have released the programming interface for adding the third-party transport protocols or rate controls.

For the OSs without interfaces for modifying its network transport layer, the second position to deploy WARC is in an application-layer library, which can support the development of streaming programs. An open-source package named *LiveMedia* library [LM07] is such a candidate. *LiveMedia* accesses the Internet via the socket interface, supports many media coding algorithms and can be used under multiple OSs, like Linux and Windows.

However, deployment in the second position has to consider whether the network conditions reported from real-time transport control protocol (RTCP) [SCF03] is enough for WARC to control the rate, where RTCP is a widely-used application-layer protocol to transmit the control messages for streaming playback. Actually, RTCP does not provide enough network conditions for the TCP-equivalent schemes

introduced in this dissertation. Therefore, an IETF draft [LG07] has been proposed to consider the required changes of RTCP to support the deployment of these schemes. The conclusion of the draft will also be applied on WARC because WARC needs the same network conditions as the existing schemes, e.g. RTT and loss events.

3.8 Summary

This work proposes a window-averaging rate control (WARC) scheme to simultaneously meet four necessary properties required by a TCP-friendly streaming-carrying scheme: fairness, smoothness, aggressiveness and responsiveness.

WARC takes the run-time estimation (RTE) model to adjust the rate. It averages the rate over a constant number of CWNDs, leading to its two inherent advantages: *fairness under stationary* conditions and *fast aggressiveness*, both are not owned by most existing well-known schemes, such as GAIMD, TFRC, TEAR, and IAD. Besides, WARC additionally employs three complements including the history-reset procedure, the one-RTT reduction procedure, and the fluid-based timeout mechanism, in order to respectively handle three special but realistic loss conditions: bursty-losses, low-multiplexing network, and heavy-losses.

The analysis shows that WARC like SIMD has the faster aggressive behavior and better tradeoffs between smoothness and aggressiveness than other schemes. That is, WARC takes shorter time to converge its rate, while providing a smoother rate at the most time. Besides, the HR procedure ensures WARC to respond to the abrupt decrease of available bandwidth right after a fixed number of packet losses. The simulation verifies the analytic results that WARC use the same bandwidth as TCP under stationary *non-periodic* losses, where most schemes fail to have that. Also, it shows that WARC uses the bandwidth close to TCP's even under the oscillating background traffic.

Briefly, while WARC uses the same bandwidth as TCP under the cases where other well-known TCP-friendly schemes cannot, WARC still provides the smoother rate than all of them. On the other hand, while WARC provides far faster aggressiveness behavior than other schemes, WARC also fast decreases its rate for an abrupt decrease of available bandwidth right after a fixed number of loss events.

Chapter 4

On Applying Fair Queuing Discipline to Schedule Requests at Access Gateway for Downlink Differential QoS

4.1 Introduction

Numerous enterprises connect to the Internet with the access link of Internet service provider (ISP), a typical topology of which could be depicted as Fig. 4.1. In general, ISPs are willing to invest money in expanding the backbone bandwidth to provide their customers better service. However, to minimize costs, their customers often delay upgrading the bandwidth of the access link, which consequently becomes the potential bottleneck to access the Internet. To guarantee key traffic getting enough bandwidth when passing through the bottlenecked link, their customers may employ a class-based fair-queuing (FQ) discipline or other packet-based bandwidth management [WTL04] at the *user-side access gateway* to scheduling packets.

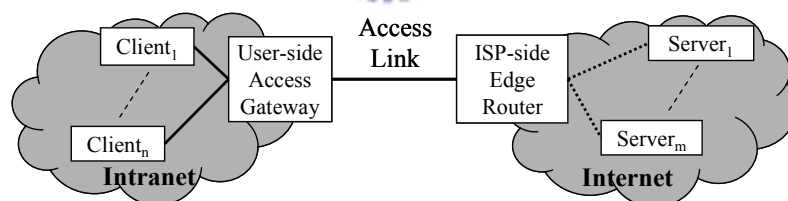


Fig. 4.1. A typical network topology that an enterprise accesses the Internet through ISP

Unfortunately, these packet scheduling solutions fail to provide such guarantee when the downlink is the bottleneck. In this case, packets are queued at the ISP-side edge router, not at the user-side gateway, for traversing the bottleneck. Scheduling packets at the user-side access gateway is useless because the packets have passed the bottleneck. On the other hand, although scheduling packets at the ISP-side edge router is useful, classifying packets at this router may be troublesome because of the network address transfer (NAT) technique, which is widely deployed at the user-side gateway to allow multiple users in an intranet sharing a public IP address. The packets which

intend to enter the intranet cannot be classified by the ISP-side edge router because the classification needs to refer to the destination IP addresses of these packets but unfortunately they have the same destination IP address before they enter the NAT-embedded user-side gateway.

Scheduling requests, instead of packets, at the user-side access gateway may solve the mentioned failure condition of packet scheduling. Such an idea is based on that applications running over the Internet mostly take the client-server model, i.e. the request/response model, such as HTTP, FTP, and E-mail. Requests sent from clients go through the access gateway and the uplink of the access link to remote servers, and the corresponding responses answered by the remote servers return to clients through the downlink of the access link and the access gateway. The bandwidth of downlink could be managed by controlling the releasing of uplink requests.

Request scheduling was used in several studies to provide differential Web QoS for different-classes users [CKD02]. These studies provided QoS services by designing request scheduling on a single Web server [PBB98, BBK00, CP99] or a web-side gateway, i.e. a gateway ahead close to a group of Web servers [CCC02, CC01, LGC01]. However, no published studies discussed how to design request scheduling at the user-side access gateway. The key difference of scheduling requests at the server or the web-site gateway from at the user-side gateway is that the target web servers in the former are specific and their statuses are easy to be measured or controlled for assisting in the scheduling operation. However, the servers in the latter are infinite, distributed over the Internet, and cannot be controlled.

In order to provide bandwidth sharing and weighted fairness among users of different classes on their downlink responses, this work studies how to schedule uplink requests at the user-side access gateway. We first investigate the possibility of applying the class-based FQ discipline, which is widely and maturely used in scheduling packets, to schedule requests. However, we found that simply applying the discipline to schedule requests would encounter three problems. The first two are related to the timing of releasing requests and the selection of the next released request. The last one is about the class-based policy, which may not suit for the user-level differentiation, i.e. may not guarantee high-class users to get more bandwidth than low-class ones when more users appear in the high class. Next, based on the above investigation, we propose a minimum-service first request scheduling (MSF-RS) scheme to provide bandwidth sharing and user-based weighted fairness, i.e.

a policy that the ratio of the bandwidth allocated for each high-class user to that for each low-class user matches the ratio of their weights.

MSF-RS consists of a minimum-service order arbiter (MOA) and a window-based rate controller (WRC). MOA always selects the next request from the class which receives the lowest amount of responses while WRC determines the timing to release a request by monitoring the downlink utilization and controlling the number of *outstanding* responses. A response is regarded as *outstanding* if its corresponding request is released, but the response has not been received completely. MSF-RS is originally designed based on the assumption that the uplink traffic consists of requests only and the downlink traffic consists of responses only. Also, it is supposed that each response comes back to answer a request forwarded by MSF-RS. However, MSF-RS also works well under the network where the exception traffic coexists with the assumed traffic. We would further discuss the traffic-mixed case and show the robustness of MSF-RS by simulation in Chapter 4.6.

The remainder of the work is organized as follows. Chapter 4.2 identifies the three problems occurring in scheduling requests with the class-based fair queuing discipline. Also, the user-based weighted fairness is introduced herein. Chapter 4.3 proposes the MSF-RS scheme. Chapter 4.4 reveals that MSF-RS does shorten the user-perceived latency and analyzes the delay bound and fairness of MSF-RS. The effects of MSF-RS on the delay and fairness are further demonstrated by the simulation results in Chapter 4.5. Besides, the affection of exception traffic on MSF-RS is discussed in Chapter 4.6. Chapter 4.7 demonstrates the effect of MSF-RS through field trail, where MSF-RS is implemented in Squid [SQI06], an open-source Web proxy package. Chapter 4.8 gives the summary.

4.2 Problems on Using Class-based Fair Queuing

Three problems would occur when the class-based FQ is used to schedule requests. The former two are related to the FQ discipline while the last is about the class-based policy.

4.2.1 The Timing for Releasing Requests

A FQ-based packet scheduler selects and sends the next packet right after the last packet has finished its transmission. The bandwidth of the link behind the scheduler would be totally consumed by the scheduled packets themselves. That is, each packet transmission can *monopolize* the link bandwidth. However, a FQ-based request

scheduler *cannot* send the next request immediately following the last request. The bottleneck downlink is consumed by responses, rather than scheduled requests.

Releasing requests one-by-one may bring a large number of concurrent response transmissions at the bottleneck downlink, because the transmission time of a response, due to its size, is often longer than that of a request. Each transmission, under such a condition, only shares small bandwidth, resulting in the serious congestion or the long user-perceived latency. On the other hand, the request scheduling cannot just wait to send the next request till the preceding request completely gets its response, because such a waiting may waste the downlink bandwidth. After a request is sent out, until its response returns, the downlink bandwidth will be idle. Besides, even when the response is transmitting, the transmission may not run out the whole downlink bandwidth.

Since requests cannot be sent out as packets, a mechanism is necessary to control the request releasing based on the bandwidth utilization, in order to avoid the downlink from congestion while keeping it on high utilization.

4.2.2 *The Determination of the Next Request*

A FQ-based packet scheduler selects the next packet which is the earliest one to be completely served, or say fully transmitted, in the fluid-based general processor sharing (GPS) model [PG93]. The order of service completion is easily determined when a packet arrives because the determination only involves two known parameters, packet arrival time and packet size. For two packets arriving at the same time, the packet size decides the order of service completion time. A smaller packet finishes service earlier.

However, in a FQ-based request scheduler, although the arrival time of each request is known, the size of a request does not affect the service time of the request, which however is counted from releasing a request to receiving its whole response and mainly contributed by the transmission time of the response. Because the transmission time is determined by the response size and the available bandwidth in Internet, it is uncertain *at the request-scheduling moment*. Therefore, the completion time is uncertain too and the request scheduling cannot serve request simply by its completion order. Hence, the selection of the next request will be a problem when the FQ discipline is applied to request scheduling.

4.2.3 *The Class-based Fairness Policy*

The class-based fairness policy is originally proposed to provide differential QoS

for different service types of connections. For example, the real-time connections and the best-effort connections would be classified into two distinct classes. Then, according to the weights of these classes, each class is allocated with a fixed proportion of bandwidth. When the class-based policy is applied, to guarantee that each connection in a class gets enough bandwidth, controlling the number of the active connections in the class is necessary. Establishing new connections will be rejected when the number of active connections exceeds the expected value.

However, when the class-based fairness policy is applied on providing differential QoS for *different levels of users*, it may expose undesirable characteristics for high-class users. For example, a high-class user may be rejected from getting service when the number of users now in the high class exceeds the expected value. Besides, if the number of users in the high class is much more than that in the low class, each high-class user may get lower bandwidth than the low-class user. Therefore, another policy may be necessary to always provide the high-class users more bandwidth particularly when more users are active in the high class than the low class. We call such a policy the user-based weighted fairness. The policy guarantees that the ratio of bandwidth allocated for each high-class user to that for each low-class user matches the ratio of their weights.

4.3 A Request Scheduling Scheme for User-side Gateway

The section proposes a minimum-service first request scheduling (MSF-RS) scheme, which is deployed at the user-side access gateway and can provide user-based weighed fairness, bandwidth sharing, full bandwidth utilization, and short user-perceived latency. As shown in Fig. 4.2, the MSF-RS scheme consists of a minimum-service order arbiter (MOA) and a window-based rate controller (WRC). The former decides which request is the next one while the latter determines the timing to release requests.

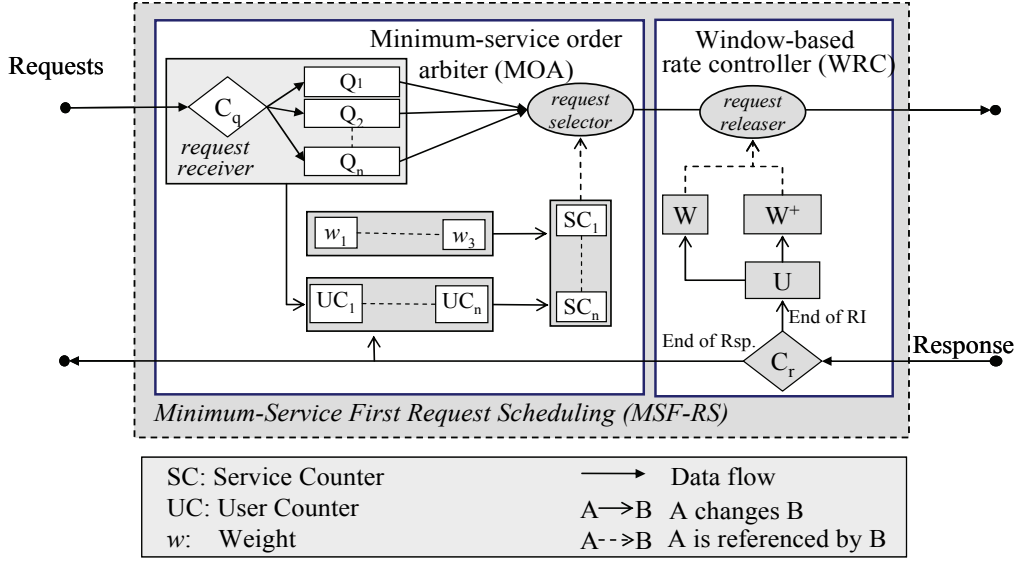


Fig. 4.2. The internal architecture of MSF-RS

4.3.1 Minimum-service Order Arbiter (MOA)

As shown in Fig. 4.2, MOA includes a request selector, a request receiver, and three groups of variables. Each class is allocated a queue Q , a user counter (UC), a service counter (SC), and a weight w .

1) **UC and SC:** The UC of a class keeps the number of the active users now in the class, where the active user means an intranet user who has requests or outstanding responses in MSF-RS. The SC of a class keeps the amount of services which the class has received. Herein the service represents the received responses in bytes after normalized with w and UC. That is, every time when one class, say the class i , partially receives its response of length L , its SC, SC_i , is updated as

$$SC_i^{new} = SC_i^{old} + \frac{L}{w_i \times UC_i}. \quad (4.1)$$

By normalizing L with w during the SC update, the ratio of the responses received by any two classes with the same SC will match the ratio of their weights. Similarly, by normalizing L with UC, the ratio of the received responses of the two classes will match the ratio of their active users. That is, even when the number of active users in the high class is much more than that in the low class, the ratio of the responses got by one high-class user to that by one low-class user still matches the ratio of their pre-assigned weights.

2) **Next request selection:** As shown in Fig. 2, when a request arrives, the classifier C_q forwards the request into the corresponding Q_i . On the other hand, the

request selector selects the head-of-line (HOL) request from the class queue with the minimum SC. A class with the smaller SC represents that it received less services than other classes. Selecting a request from such a class queue improves the fairness between classes, because it minimizes the difference of their SCs, i.e. on their received services. Besides, if multiple classes have the same SC, the request selector would select the class with the highest product of the weight and UC.

An idle class represents that the class has no outstanding responses and its request queue is empty. When a class idles for a long period, its SC may be far smaller than the SCs of other classes with backlogged requests and response. Once the idle class has incoming requests, its far-small SC may cause the starvation of other classes. That is, until its SC is larger than any one SC of other classes, no request can be selected from other classes. Such a condition may be unfavorable. To avoid the condition, once the idle class becomes busy, its SC is updated to the minimal SC among all active classes. Such an update lets MOA follow the sharing concept used in the fair queuing discipline: the bandwidth freed from the idle classes would be shared by active classes, and these active classes will not get less bandwidth because of the sharing when the idle classes become active. Notably, if all classes are idle, all SCs will be reset to 0.

3) *Basic Procedures*: Fig. 4.3 lists the pseudo codes for the two components in MOA. The *request_selector* picks the class queue with the minimum SC and releases the HOL request of this queue. The *request_receiver* classifies and en-queues all incoming requests. If the arrival request is classified into an idle class, i.e. the class's UC is zero, the *request receiver* reset the SC of this class. Next, the request is put into the specific queue, according to its class. If the request comes from a user j who has no request waiting for service or being served, i.e. $ReqFromUser[j]=0$, then the UC of its class will be added one, implying one more user arrives in the class. If the system is idle, i.e. no responses are outstanding, the *request_receiver* actively asks the *request_scheduler* to release the just coming request immediately.


```

Array ReqFromUser[j] : # of requests now in MOA and
                        coming from the jth user

boolean Request_Selector // Called by Request_Receiver or WRC
{
  // Select the queue with min SC among all active queues,
  // i.e. their corresponding classes' PendingPpt are not zero.
  // return null if no active queues.
  Qi ← ActiveQ_with_minSC()
  If (Qi = null)
    return False // imply there are no active classes
  Else
    SendHeadReq(Qi)
    return True
}
Procedure Request_Receiver(Req) // Called when a request arrives
{
  i ← GetClassNo (Req)
  j ← GetUserNo(Req)
  If (UCi≠0)
    SCi←GetMinSC()
  Enqueue(Req)
  If (ReqFromUser[j]=0)
    UCi= UCi+1
  ReqFromUser[j]++;
  If (W = 0)
    Request_Selector ()
}

```

Fig. 4.3. Two procedures in MOA: request selector and request receiver

4.3.2 Window-based Rate Controller (WRC)

As shown in Fig. 4.2, WRC controls the maximum of outstanding responses, W^+ , according to the bandwidth utilization of the link, denoted by U . The variable W is used to record the current number of outstanding responses. The U is updated by the expression

$$U_k = \frac{S_k / T}{C}, \quad (4.2)$$

where U_k is the utilization between the k^{th} and $(k+1)^{\text{th}}$ updates, C is the link capacity, T is the time interval between two updates, and S_k is the responses in bytes received during T . Next, once U_k is changed, WRC updates W^+ by the equation

$$W_{k+1}^+ = \min\left\{\frac{U^+}{U_k}, K\right\} \times W_k^+, \quad (4.3)$$

where W_{k+1}^+ is the maximum of outstanding responses allowed after the $(k+1)^{\text{th}}$ update and $W_0^+ = 1$. U^+ is the expected utilization. When U_k is lower than U^+ , W_{k+1}^+ will be set to a larger value so that more outstanding responses can use the bandwidth and then raise the utilization U_{k+1} . For example, if the current W^+ is 6 and U is 60%, then the next W^+ will be set to 10 when $U^+=99\%$. On the contrary, when U_k is

higher than U^+ , W_{k+1}^+ will be decreased so that fewer responses compete for the downlink bandwidth.

The U^+ is constant and should be smaller than 100% ($U^+ < 100\%$). Otherwise, when $U^+=100\%$ and $U_k=U^+$, it cannot be distinguished whether the bandwidth required by the responses is larger than or just equal to that of the link. K is a constant and assigned to 2 to avoid WRC from over-estimating the new W^+ particularly when the old W^+ is small. Notably, W^+ should be recomputed only when $W=W^+$. When $W < W^+$, it is wrong to expect the raise of U by increasing W^+ , because the low U is caused by the insufficient arrival requests, but not the insufficient W .

Fig. 4.4 lists the pseudo code of WRC. When WRC receives any part of a response, it looks for the class and the user which this response belongs to, and updates its class's SC. Once the received data includes the last packet of a response, W is decreased by one to imply a request having been fully answered. Also, ReqFromUser of this user is decreased one. If this is the last request, UC of this class is decreased one also, because one user leaves the class. Next, the *request_scheduler* is invoked to release requests as more as possible, till $W=W^+$ or all request queues are empty.

```

Procedure WRC // called when partial response returns
{
  Data ← GetData()
  i ← GetClassNo(Data)
  j ← GetUserNo(Data)
  Len ← Size(Data)
  SCi ← SCi + Len / wi // update SC by (1)

  If (IsTail_of_Rsp(Data)) { // ending event of a transaction
    W ← W - 1
    ReqFromUser[j] ← ReqFromUser[j] - 1
    If (ReqFromUser[j]=0)
      UCi ← UCi - 1
  }

  While (W < W+) // imply more transactions are expected
  {
    If (Request_Scheduler()=False) // ask for releasing a request
      Goto no_reqs // if all req queues are empty
    W ← W + 1
  }
Label no_reqs:
}

```

Fig. 4.4. Procedure of window-based service-rate controller (WRC)

4.4 Analysis for Delay and Fairness

In the section, we first demonstrate that a MSF-RS gateway provides users shorter latency than an ordinary gateway on the average. An ordinary gateway means

that it directly forwards the requests or responses once receiving them from client hosts or destination sites, respectively. Then, we analyze the delay and fairness provided by MSF-RS in the worse case.

4.4.1. Short User-perceived Latency

Two definitions are given for conveniently analyzing the user-perceived latency.

Definition 4.1: A *transaction* includes a request and its corresponding response. Besides, a transaction queued in a gateway G means that the request of the transaction is queued in G . Moreover, a transaction being served by a gateway G represents that G forwards the request of the transaction to the destination site, and later receives its response from the destination site while forwarding it back to the client. Herein we assume that a gateway can receive a response from the destination site while forwarding it to the client host. Besides, the response of the transaction in serving is also called an outstanding response in this work.

Definition 4.2: The active time of a transaction in a gateway G begins when G receives the request of the transaction and ends when G returns the whole response of the transaction back to the host. If T_a denotes such active time, then T_a consist of the queuing time (T_q) and the service time (T_s) of the transaction, according to Definition 1. That is, $T_a = T_q + T_s$. Also, because T_s is dominated by the transmission time of the response, we simply define T_s is a function of L and b , where L is the length of the response and b is the average bandwidth used by the response during its transmission.

The user-perceived latency of a request begins when the client host sends out the request and ends when it receives the whole responses of the request. That is, the latency includes the data transmission time between the client host and the gateway G , and between G and the destination site. However, the time in the former part is eliminated from the following comparison because it approximates a small constant no matter MSF-RS is deployed or not in the gateway. Therefore, we would show that a MSF-RS gateway could provide shorter time in the latter part than an ordinary gateway on average.

According to Definition 4.2, T_a just represents the time in the latter part. The following compares the MSF-RS and ordinary gateways under a case that a batch of m requests arrives into the gateways. Assume MSF-RS has a fixed W^+ . Besides, the maximum bandwidth that W^+ responses can use would approximate to the downlink bandwidth C , because if a MSF-RS gateway allows W^+ responses concurrently

transmitting, then these responses are expected to completely occupy but not overload the downlink. Therefore, the maximum bandwidth of each response can be expressed as C/W^+ .

Let us first consider the situation under an ordinary gateway. Because the ordinary gateway directly forwards any received requests, the T_q of these transactions is closed to zero, compared to their T_s . Besides, since the responses of the m requests will concurrently share the downlink bandwidth, the bandwidth got by each response can be written as $\frac{C}{m}$. Therefore, the average T_a of transactions under an ordinary gateway is

$$\text{avg}(T_a^{\text{ordinary}}) = 0 + \frac{L}{\frac{C}{m}} = \frac{mL}{C} \quad (4.4).$$

Next, consider the case under MSF-RS. The T_q 's of the first W^+ transactions are zero because their requests are forwarded immediately. Then, others transactions will be queued until any of the W^+ transactions have been served. In the worst case, these transactions end concurrently. Thus, the T_q 's of the next W^+ transactions would equal to the T_s of the first W^+ transaction, i.e. $\frac{W^+L}{C}$. Next, the T_q 's of the residual transactions could be derived from the same way. Thus, by summing up the T_q 's of W^+ transactions in each round and considering the possibility that the number of transactions in the last round may be less than W^+ , the average T_q of the m requests could be expressed as

$$\text{avg}(T_q^{\text{MSF-RS}}) = \frac{1}{m} \left(W^+ \left(0 + 1 + 2 + \dots + \max \left\{ \left\lfloor \frac{m}{W^+} \right\rfloor - 1, 0 \right\} \right) + (m \bmod W^+) \left\lfloor \frac{m}{W^+} \right\rfloor \right) \frac{W^+L}{C}. \quad (4.5)$$

Similarly, the mean T_s of the m transactions could be expressed as

$$\text{avg}(T_s^{\text{MSF-RS}}) = \frac{1}{m} \left((m - (m \bmod W^+)) \frac{W^+L}{C} + (m \bmod W^+) \frac{(m \bmod W^+)L}{C} \right). \quad (4.6)$$

Therefore, we get the average T_a of transactions under a MSF-RS by summing up Eq. (4.5) and Eq. (4.6).

To compare the average period of transactions over the two gateways, Fig 4.5 plots the ratio of the MSF-RS gateway to an ordinary gateway on T_a over different m and W^+ . Fig. 4.5 shows that the ratio is smaller than 1 always, i.e. the average time to finish transactions under MSF-RS is no more than that under an ordinary gateway. For example, as plotted by the dotted line, MSF-RS can reduce 25% of T_a when the

number of arrival requests is two times of W^+ . These results demonstrates that MSF-RS does not cause additional delay on T_a through MSF-RS does queue requests, i.e prolong T_q , to provide differential services for different classes of users.

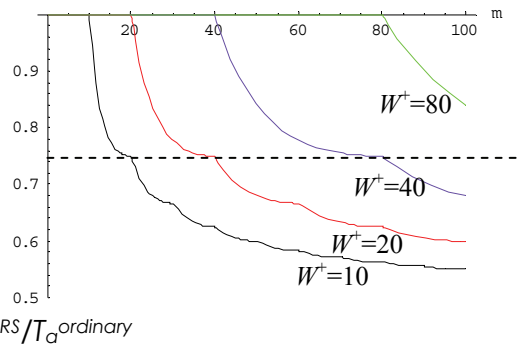


Fig. 4.5 The ratio on T_a of a MSF-RS gateway to an ordinary gateway

4.4.2. Delay Bound

Two models are introduced before the definition and analysis of delay bound. As shown in Fig. 4.6(a), we model MSF-RS as a multi-links fair queue to analyze the delay bound caused by MSF-RS in the worse case. Assume that W^+ is fixed and each class has equal users in the residual analysis. The downlink with capacity C is conceptually divided into W^+ sub-links and each sub-link has bandwidth $c=C/W^+$. Besides the model for MSF-RS, we describe an ideal MSF-RS gateway, named MSF-RS*, as shown in Fig. 4.6(b). Under MSF-RS*, the downlink bandwidth C can be *proportionally* shared by N classes according to their weights. For example, if three classes of users receive their responses through the downlink and the ratio of their weights is 4:2:1, then the bandwidth C should be divided into 3 sub-links which bandwidth are $4/7C$, $2/7C$, and $1/7C$, respectively. Besides, the bandwidth got by each class would be dynamically adjusted according to the number of non-empty class queues.

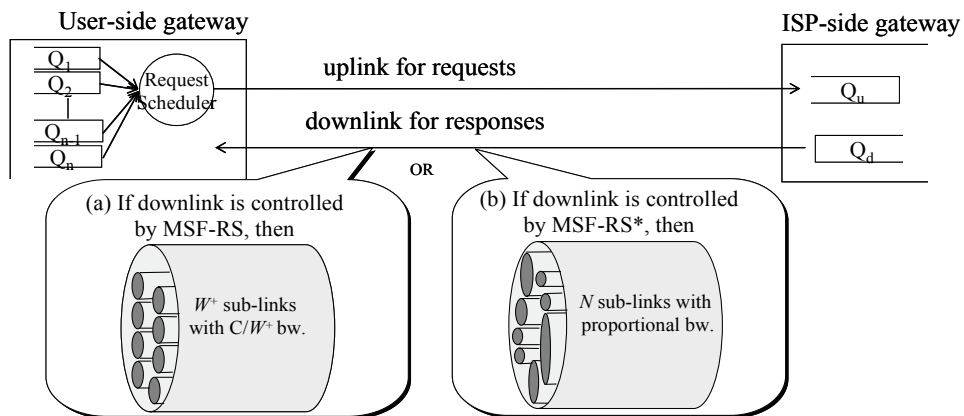


Fig. 4.6. The downlink can be conceptually divided into multiple sub-links.

Definition 4.3: An active class represents that the class has transactions handled in the gateway. Also, an active period of Class i is the time during Class i is active.

Definition 4.4: A backlogged class represents that the class has a non-empty queue of requests.

Definition 4.5: The delay bound for a transaction in a MSF-RS gateway means $T_a^{\text{MSF-RS}} - T_a^{\text{MSF-RS}^*}$, i.e. the additional time caused by MSF-RS, compared to an ideal MSF-RS gateway.

Such a delay may be contributed from two parts. The first is the queuing time of requests in the MSF-RS gateway. For the first request queued in a class, its queuing time under a MSF-RS^{*} gateway is zero because MSF-RS^{*} has $W^+ = n$ and thus the request must be immediately forwarded. However, the request may be delayed in MSF-RS when all sub-links are busy. Another source to contribute the delay is the additional transmission time of responses if lower downlink bandwidth is allocated for a class than its allocated bandwidth, which is possible in MSF-RS. Since a MSF-RS gateway only affects the allocation of downlink bandwidth by controlling the releasing of uplink requests, it cannot directly adjust the downlink bandwidth used by each class.

Assume that a_k denotes the time of a request k arriving at MSF-RS, b_k denotes the time it departed from MSF-RS, and d_k denotes the time that MSF-RS finished to return the whole response back to the client host. Similarly, b_k^* , and d_k^* denotes the corresponding times at MSF-RS^{*}. The denotation a_k^* is unnecessary since the arrival times at MSF-RS and MSF-RS^{*} are equal. Next, assume the total length of the response is L_k and $d_k = b_k + \frac{L_k}{r}$ where r represents the receiving rate of the response. Herein, the time between b_k and receiving the first data of its response is ignored, since it is far smaller than the receiving time of the whole response.

Theorem 4.1: The delay in MSF-RS has a bound equal to

$$\frac{2L^+}{c},$$

where L^+ is the maximum responses among all classes and c is the bandwidth of a sub-link. That is, for any request k , the time that MSF-RS finishes its transaction must not be $\frac{2L^+}{c}$ later than that under MSF-RS^{*}, i.e. $d_k - d_k^* \leq \frac{2L^+}{c}$.

Proof:

Case 1: The request k arrives when $W < W^+$.

In this case, $b_k = a_k$ because MSF-RS immediately sends out requests if $W < W^+$. Then, $d_k = a_k + \frac{L_k}{c}$. Besides, $d_k^* \geq a_k + \frac{L_k}{C}$ because the capacity C of a link is the maximum bandwidth that a transaction can use in MSF-RS*. Thus,

$$d_k - d_k^* \leq \frac{W^+ - 1}{W^+} \frac{L_k}{c} \quad (4.7)$$

Case 2: The request k arrives when $W = W^+$.

Consider the transaction of the request k is the $(m+1)^{\text{th}}$ transaction in the same active period of Class i . Let us mark the transaction. Next, we calculate how many transactions will be served from other classes ahead of this marked transaction in the worst case. Assume n is the number of active classes now in the gateway and all W^+ sub-links are serving transactions. Assume these transactions have responses of the largest size L^+ . That is, in the worst case the earliest time to release the next request from the gateway will be at $a_k + \frac{L^+}{c}$. Next, for each Class j where $j \neq i$, it may receive a maximum $\frac{mL_i^+}{r_i} \times r_j$ of total responses, before their timestamps are assigned to a value equal to that of Class i , where L_i^+ denotes the maximum response of Class i . Moreover, as mentioned in Chapter 4.3.1.2, when all classes have the same timestamp, MSF-RS forwards requests from the class with the highest weight. Thus, if Class i has the q^{th} highest weight, then its request will be forwarded at the rank of q among all classes with the same timestamp. Therefore, the time at which the marked transaction will be finished is given by

$$d_k \leq \left(a_k + \frac{L^+}{c} \right) + \frac{1}{W^+ c} \left(\sum_{j=k-m}^{k-1} L_j + \sum_{\substack{j=1 \\ j \neq i}}^{n-1} m \frac{L_i}{r_i} r_j \right) + \left\lceil \frac{q}{W^+} \right\rceil \frac{L^+}{c}. \quad (4.8)$$

However, under MSF-RS* the request k will be forwarded once no requests queued in Class i . Besides, because a class with the q^{th} highest weight must get no more bandwidth than the classes with weight equal to or higher than it, it can get at most $\frac{W^+ c}{q}$ bandwidth. Therefore, the time at which the marked transaction will be finished under MSF-RS* is given by

$$d_k^* \geq a_k + \frac{1}{W^+c} \left(\sum_{j=k-m+1}^{k-1} L_j + \sum_{\substack{j=1 \\ j \neq i}}^{n-1} m \frac{L_i}{r_i} r_j \right) + \frac{L^+}{W^+c}. \quad (4.9)$$

Therefore, by subtracting Eq. (4.9) from Eq. (4.8) and considering Eq. (4.7), we get the delay bound under MSF-RS is given by

$$d_k - d_k^* \leq \frac{L^+}{c} + \left(\left\lceil \frac{q}{W^+} \right\rceil - \frac{q}{W^+} \right) \frac{L^+}{c} \leq \frac{2L^+}{c}.$$

4.4.3. Fairness

The fairness parameter that we use is based on the definition presented by Golestani [GOL94] for the analysis of self-clocked fair queuing. The parameter is defined as the maximum difference of the service got by any two backlogged classes over arbitrary time intervals. A scheduling algorithm has a zero value of fairness if it always provides equal service for any two classes even in a short time interval. The fairness of MSF-RS* is zero since the downlink bandwidth can be concurrently shared by all non-backlogged classes anytime. The following analyzes the fairness of MSF-RS.

Theorem 4.2. The fairness of MSF-RS is

$$\frac{W^+L_i^+}{w_i} + \frac{L_j^+}{w_j},$$

where i and j could be the indexes of any two backlogged classes and $\frac{L_i^+}{w_i} \geq \frac{L_j^+}{w_j}$.

Proof: We consider two classes, Class i and Class j , through the following analysis because the definition of fairness only concerns two classes. The existence of more active classes does not affect the difference of services got between two classes. Besides, let $\frac{L_i^+}{w_i} \geq \frac{L_j^+}{w_j}$, where w_i and w_j are the weights of Class i and j , respectively. As

shown in Fig. 7, assume the MSF-RS is idle before the time t_0 , i.e. $W=0$. Then, at the time t_0 more than W^+ requests of Class i arrive. Let v denotes the timestamp of the first request of Class i , where timestamp represents the value in SC_i when the request arrives into Class i . Upon these requests arrives, MSF-RS will forward the first W^+ ones and W is set to W^+ . Let k be the $(W^++1)^{\text{th}}$ request, i.e. it would be the first request in the queue of Class i after t_0 . Since W^+ requests of Class i would be served before k ,

k will have a timestamp v' expressed by

$$v' = v + \frac{1}{w_i} \sum_{h=1}^{W^+} L_i^h \leq v + \frac{W^+ L_i^+}{w_i}.$$

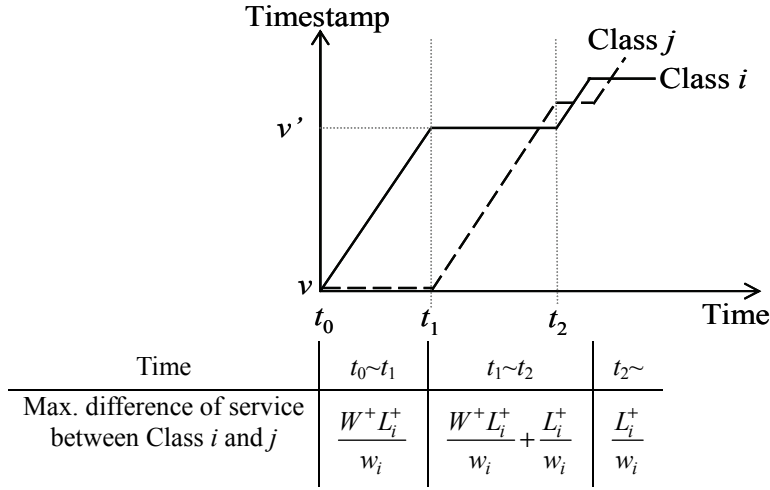


Fig. 4.7. The difference of the service between Class i and Class j

Assume that requests of Class j arrive right after the time t_0 , and the first request should have the timestamp v since the timestamp of the request latest released by the server is v . However, although the first request of Class j has timestamp v smaller than that of Class i , i.e. v^k , no requests can be forwarded from Class j because all W^+ sub-links are busy for Class i . Let the sub-links become idle at t_1 . Then, in the worse case all requests of Class j with a timestamp *no larger* than v' will be forwarded before k . Assume the Class- j request with v' asks a response with size equal to L_j^+ . Then, between t_1 and t_2 the maximum of the total responses received by Class j will be $w_j \left(\frac{1}{w_i} W^+ L_i^+ \right) + L_j^+$. However, as shown in Fig. 4.7, Class i does not get any service during the period. Thus, between t_1 and t_2 , the difference of the service between the two classes is $\frac{W^+ L_i^+}{w_i} + \frac{L_j^+}{w_j}$. For the time later than t_2 , when the service of a class already equals to another one, the additional service which the class can get must smaller than $\frac{L_i^+}{w_i}$ since $\frac{L_i^+}{w_i} \geq \frac{L_j^+}{w_j}$ is supposed. Since the difference of service after t_2 and before t_1 is smaller than that between t_1 and t_2 , the difference of the service between t_1 and t_2 is the fairness of MSF-RS, i.e. $\frac{W^+ L_i^+}{w_i} + \frac{L_j^+}{w_j}$.

4.5 Simulation Results

This section verifies the effects of MSF-RS by *ns-2* [NS06] in terms of the fairness and -bandwidth sharing, user-perceived latency, the relationship between U and W^+ , the effect of U^+ on latency.

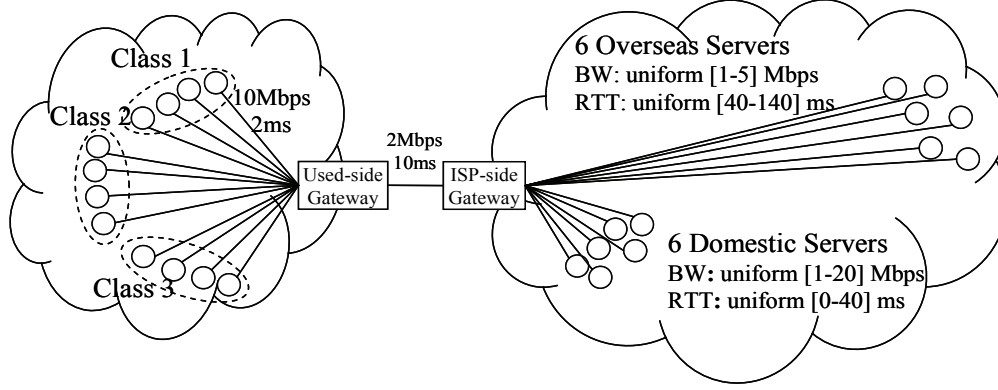


Fig. 4.8. Simulation topology for three classes with service ratio 4:2:1

4.5.1 Topology

The *HTTP/Cache* in *ns-2* acts as a web proxy cache, and sits between clients and web servers. It intercepts the requests sent from clients and forwards them to the remote servers if the requested data is not cached yet. This work disables the cache function and implements MSF-RS in *HTTP/Cache*. Fig. 4.8 shows the topology used in the simulation. The MSF-RS gateway provides three classes, Class 1, Class 2, and Class 3, with the weights, 4, 2, and 1, respectively. Each class involves four clients and each client repeatedly requests pages from the 12 remote web servers through the MSF-RS gateway. For each client, its time interval between two requests is an exponential distribution with mean equal to 5 seconds. The link between the MSF-RS gateway and every client is 10Mbps with 2ms propagation delay. The ISP router connects to twelve servers with twelve independent links. These servers are classified into two equal numbers of groups, representing overseas servers and domestic servers. Links between the gateway and these servers have a *uniform* distribution, as shown in Fig. 4.8. By the statistics from the real Internet [BC98], the web response size has a lognormal distribution with $M=9.357$ and $S=1.318$. The average response size is $e^{M+S^2/2}$ bytes, i.e. 27,656 bytes. The U^+ in WRC is set to 98% and the time interval between two updates is set to 5 seconds. Chapter 4.5.6 would further reveal the effects of different U^+ 's on link utilization, but that of different update intervals are ignored

to show because of their insignificant effects. Besides, we use TCP SACK and assume no delayed acknowledgments. Over the simulation the packet size is 1000 bytes and the maximum congestion window of TCP is 200. The queues at the two gateways are managed by Drop-Tail and their sizes are 1.5 bandwidth-delay products.

4.5.2 Weighted Fairness and Bandwidth Sharing

First, we demonstrate that when all classes have the same users, MSF-RS provides weighted fairness between classes and the idle bandwidth is shared by active classes. Four phases are included in the simulation and the duration of each phase is 200 seconds. In the first phase, all of the three classes have backlogged requests. In the next two phases, Class 1 and 2 stop requesting individually, and then both of them have backlogged requests again in the last phase.

Fig. 4.9 shows the average throughput in each phase. During the first phase, the three classes get proportional bandwidth in ratio 3.96:1.98:1, which is close to the expected ratio 4:2:1. In the second phase, the idle bandwidth freed by Class 1 is shared by Class 2 and Class 3 proportionally. Both of the bandwidth obtained by Class 2 and Class 3 increase in this phase, and the usage ratio between them is still 2:1. After Class 2 stops requesting in the third phase, Class 3 occupies all bandwidth until the end of this phase. During the second and the third phases, Class 1 and Class 2 still obtain a bit of bandwidth separately due to their unfinished transactions at the 200th and 400th second, respectively. Once all idle classes have requests again in the last phase, the three classes obtain the bandwidth in the expected proportion, 4:2:1, again.

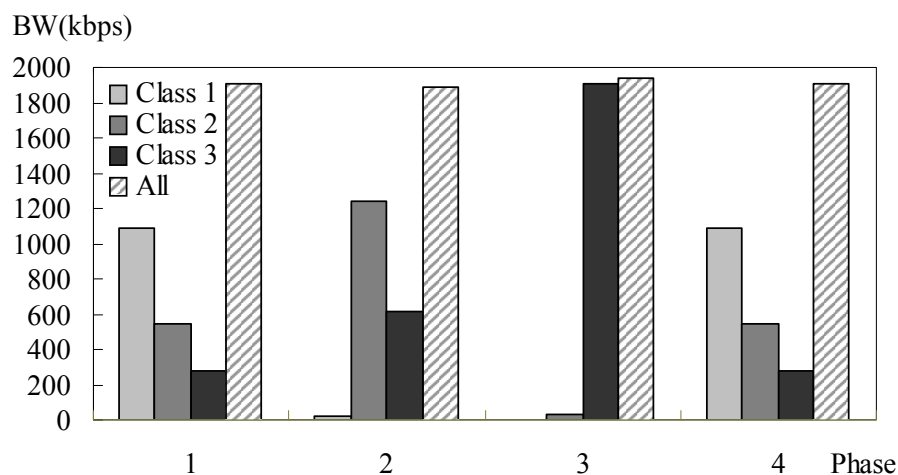


Fig. 4.9. The average throughput of three classes over the four phases

As mentioned in Chapter 4.1, packet scheduling algorithms fail to allocate the

downlink bandwidth at the user-side gateway, because the packets of responses have passed the bottleneck and can be immediately forwarded to the clients. Under the situation, a packet scheduling algorithm would degrade into a first-in-first-out scheduling. To demonstrate such degradation, we employ a deficit round robin (DRR) [SV96] instead of MSF-RQ at the user-side gateway. DRR is a widely used packet scheduling algorithm because it is easy to implement. Fig. 4.10 shows the bandwidth allocation managed by DRR. Obviously, over the four phases, Class 1 and Class 2 do not get higher bandwidth than Class 3, even though both classes have larger weights than Class 3. Further observation shows that the request queues of three classes in DRR are empty during the simulation, which verifies that requests will be forwarded upon their arrival, i.e. forwarded with a FIFO order.

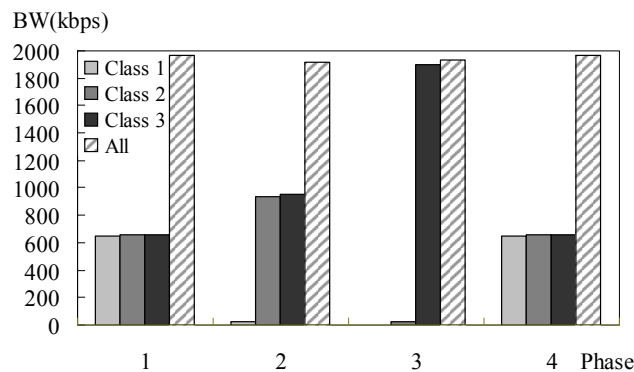


Fig. 4.10. The average throughput of three classes over the four phases under DRR

4.5.3 User-perceived Latency

The simulation scenario here is the same as that used in the first phase in Chapter 4.5.2. Fig. 4.11 illustrates the user-perceived latency for the three classes, the average latency of all classes, and the latency if no MSF-RS is deployed, denoted as non-MSF-RS. Also, the latency is decomposed into the queuing time and the service time of the transactions.

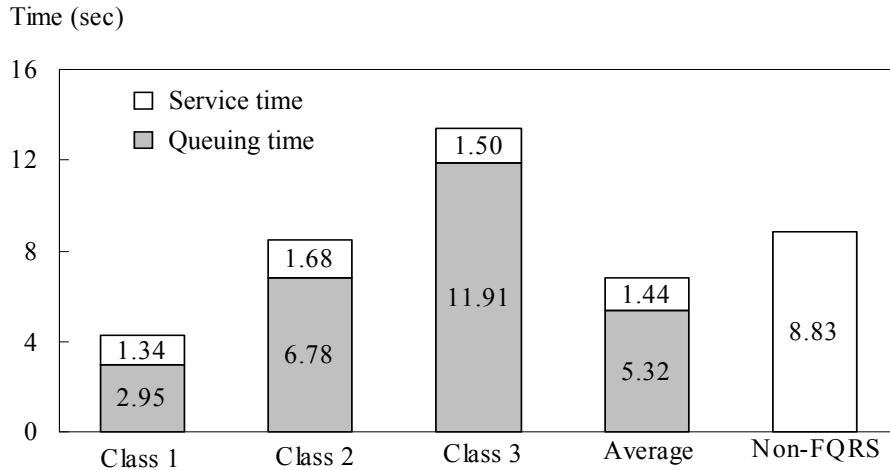


Fig. 4.11. User-perceived latency comparison by decomposing time factors: queuing time and service time

First, by comparing the left three bars, the three classes in MSF-RS experience the different user-perceived latencies, mainly caused from different queuing time since they have different weights. Second, by comparing the right two bars, the average latency (6.76 secs) in MSF-RS is shorter than that in non-MSF-RS (8.83 secs) by 23.44%. It is because the average service time in MSF-RS (1.44 secs) is far shorter than that in non-MSF-RS (8.83 secs). The service time in MSF-RS is reduced because of the well-controlled number of concurrent outstanding responses.

4.5.4 User-based Weighted Fairness

Next, we show the MSF-RS gateway provides the high-class users more bandwidth than the low-class users regardless of the number of users in the high class. The same testing scenario as that in Chapter 4.5.2 is used, but the number of users in Class 1 is increased in each test from 4 to 24. Also, all of the three classes have backlogged requests during the whole testing time, 800 seconds. Fig. 4.12(a) plots the difference of the average bandwidth allocated for the users in each class. When there are 4 users in Class 1, each user in this class owns 270Kbps, which is the two and four times of that allocated for the user in Class 2 and Class 3, respectively. The fixed ratio on the allocated bandwidth among the three classes is kept even when users in the first class are increased. Fig. 4.12(b) plots the result provided by the MSF-RS gateway without considering the number of users when updating SC, i.e. the L in Eq. (4.1) is not divided by the UC . Obviously, under such a gateway, the users in Class 1 cannot be ensured to get more bandwidth than that in other classes when more users are active in Class 1.

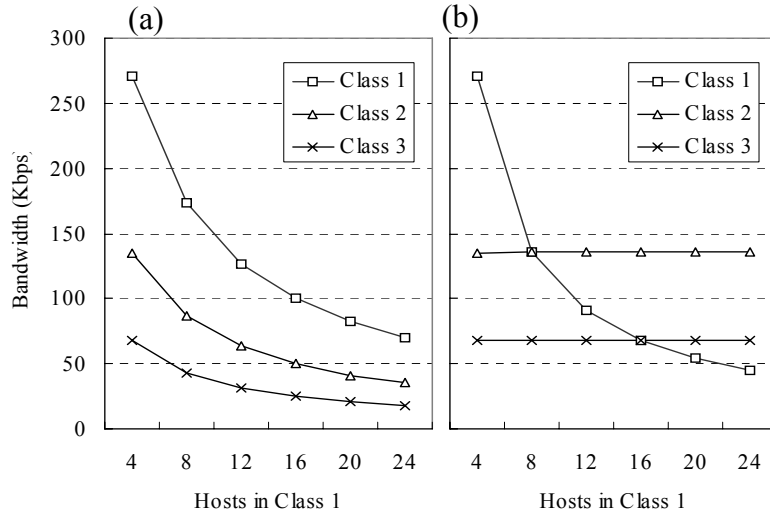


Fig. 4.12. The difference on the bandwidth allocated for the high-class host between the host-based and class-based weighted fairness.

4.5.5 Adjustment of Outstanding Responses

The subsection observes the adjustment of W^+ when the arrival of requests is not backlogged always, i.e. the scheduling server may be idle sometimes. In the 1500-second test, clients in Class 1 and 2 send requests every 1 second while that in Class 3 send one every 10 second. Besides, during the middle 500 seconds, clients in Class 1 and 2 stop sending requests, which results in insufficient requests so that the MSF-RS gateway has no request to send out. Fig. 4.13 reveals the relation between U and W^+ . The utilization of access link stays around 0.98 as the expected U^+ in the first and last 500-second periods because of sufficient arrival requests.

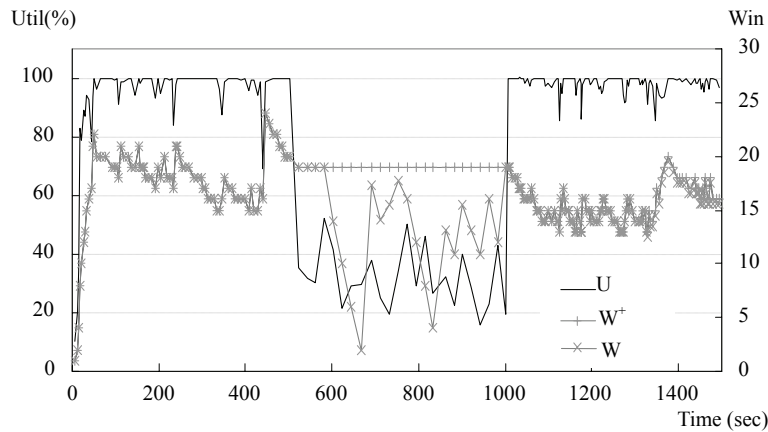


Fig. 4.13. The size of W^+ is fixed in the period with insufficient traffic (the 500th ~1000th seconds)

In the 500th~1000th sec, the utilization falls apparently and the value of W^+ keeps constant as described in Chapter 4.3.2. Increasing W^+ for raising the utilization during the period is in vain because the low utilization results from the fact that the incoming

requests are too few to occupy all sub-links. Besides, the value of W varies with a wide range, determined by the dynamically arrival requests. During the period, any requests are forwarded immediately once they arrive, since there are always free sub-links. Notably, at the 1000th second, once the two stopped classes restart sending requests, all requests can be released soon and the utilization jumps to the expected value.

4.5.6 Effect of U_{max} on Latency

Fig. 4.14 depicts the user-perceived latency, the queuing time spent in the MSF-RS access gateway and in the ISP-side edge router when U^+ is assigned from 0.65 to 0.99. Raising U^+ follows shorter user-perceived latency, because more responses can be concurrently transmitted and the bandwidth can be more utilized. However, the raise also causes packets to be queued in the ISP-side router because of less free bandwidth for eliminating the queued packets as U^+ is high. By the observation in Fig. 4.14, the value of U^+ is suggested to be set to 98%.

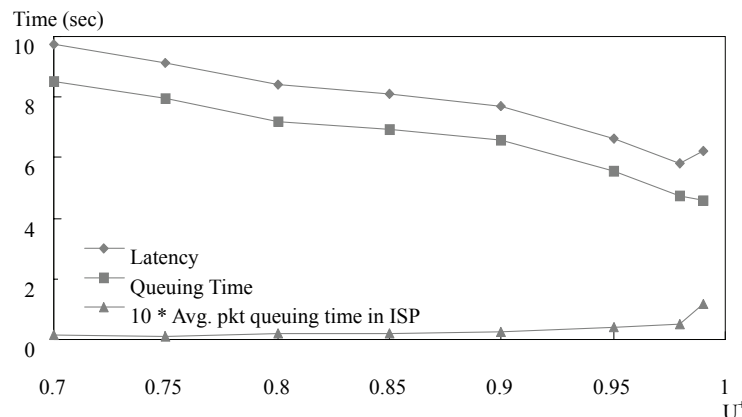


Fig. 4.14. The user-perceived latency, queuing time, and the number of packets queued in ISP-side router under different U^+

4.6 Affection of Exceptive Traffic

MSF-RS is designed under the assumption that the uplink traffic comprises requests only and the downlink traffic comprises their corresponding responses only. However, the exceptive traffic does coexist with the assumed traffic in the real environment. We classify the exceptive traffic into three types and explain why they do not affect the fairness or link utilization provided by MSF-RS.

1) *Uplink exceptive traffic*: The type of traffic may include the uplink responses and the packets actively sent from the internal users. If this traffic is heavy enough to turn the uplink to a bottleneck, a packet scheduler with the FQ discipline is suggested

to be deployed at the access gateway first. MSF-RS can coexist with the uplink FQ discipline well, as shown in Fig. 4.15(a). Also, if the weighted fairness on uplink is not a concern, the combination of MSF-RS and priority queues is a simple solution, as shown in Fig. 4.15(b). The solution gives the request traffic higher priority since they are smaller than responses usually.

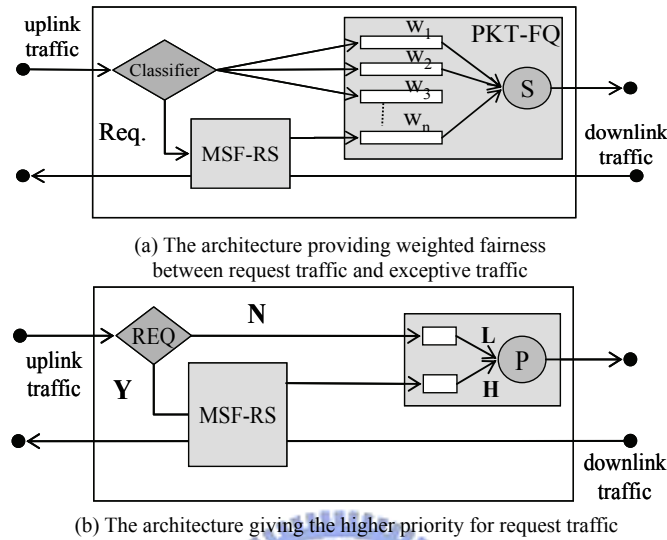


Fig. 4.15. Two potential integrated architectures for handling the network when the uplink is a bottleneck.

2) *Downlink excessive traffic belonging to some classes*: Such traffic is still the downlink responses, but their requests are not recognized by the implementation of MSF-RS. For example, POP3 mail traffic for someone's host belongs to Class i . It is possible since only the web request is recognizable for the present implementation of MSF-RS.

The MOA in MSF-RS regards these excessive packets as the received service of classes. That is, when the packets not triggered by an (recognized) request arrive from the Internet, their sizes are accumulated into the SC of the class which the packets belong to, as other response packets triggered by requests. That is, if a user of Class i receives a crowd of such packets from the Internet, the sizes of packets would be accumulated into SC_i . Although the additional value in SC_i caused by these excessive packets brings the fewer requests sent out from Class i by MSF-RS, it does not affect the weighted fairness between classes and the link utilization.

3) *Downlink excessive traffic not belonging to any class*: Excessive traffic not belonging to any class may include requests from outside network for the responses provided by the internal servers, or the malicious attacks. The former case is possible for the enterprises having web servers for their customers. The excessive traffic

contributed from such requests is usually small, compared with other response traffic running on the downlink. The latter case may rudely and fully occupy the downlink, resulting in the failure for all transmissions. However, the latter case is a security problem and out of the scope of this work.

Actually, this exceptive traffic could be considered as response traffic, when WRC monitors the utilization of the downlink to set W^+ . Such traffic would bring a smaller W^+ than that in the case without the exceptive traffic. That is, WRC may think such a small W^+ is enough to fully utilize the downlink bandwidth. Besides, when the exceptive traffic passed, because WRC would quickly adjust W^+ according to the new U , the link utilization is not affected in this case.

We use the following simulation to demonstrate that the utilization is not affected by the downlink exceptive traffic. The on-off CBR traffic not belonging to any classes with different rates during different periods are generated to demonstrate the responsiveness of WRC is fast enough to keep the high link utilization. Five on/off periods are tested: 40, 80, 160, 320, and 640 seconds. During the on-period, three rates of CBR traffic are tested individually: 20%, 40%, and 60% of the downlink capacity. Each test is run for 3000 seconds and the U^+ is set to 98%. Fig. 4.16 shows the case where on/off period is 640 seconds and CBR rate is 40% of link capacity, i.e. 0.8 Mbps. At 1280 and 2560, once the CBR traffic stops, WRC immediately resets W^+ from 7 to 13 in order to release more requests. The bandwidth freed by CBR traffic is fully and fast occupied by the response traffic. In fact, the results in all tests, as shown in Table I, reveal that WRC keeps the utilization on 97.84% averagely, closing to the designed goal, 98%.

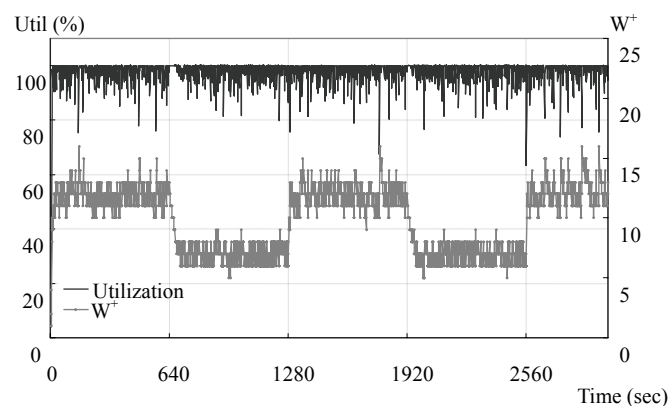


Fig. 4.16. Fast-responsive W^+ and full utilization of access link under oscillate CBR traffic

TABLE 4.1. THE UTILIZATION OF LINK
UNDER OSCILLATING CBR TRAFFIC ($U^+=98\%$)

On/Off Period	The Rate of CBR During On-period		
	0.4 Mbps	0.8 Mbps	1.2 Mbps
40 sec	97.94%	97.91%	97.62%
80 sec	97.90%	97.95%	97.77%
160 sec	97.94%	97.92%	97.68%
320 sec	97.83%	97.80%	97.76%
640 sec	97.93%	97.83%	97.85%

4.7 Field Trial

We implemented MSF-RS in *Squid* [SQI06], which is an open source package of web proxy cache, and performed field trial in an open network environment. Fig. 4.17 illustrates the test bed for evaluating the MSF-RS in Squid. An application-layer traffic generator named *Avalanche* [AVA06] is used to emulate the behaviors of multiple clients and send requests to the web servers in the Internet. *Avalanche* is imported with a URL list, a historical record logged by an enterprise in a couple of days.

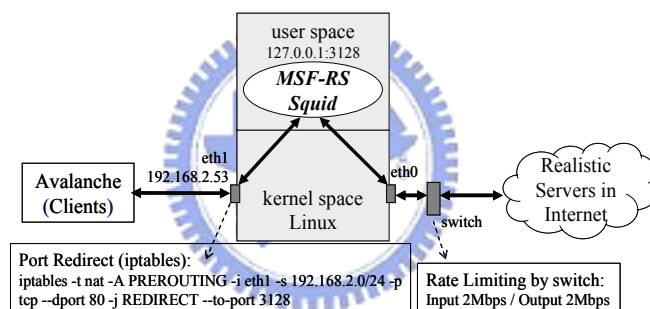


Fig. 4.17. The test bed for field trial in the Internet

The access gateway installed with MSF-RS is configured as a transparent proxy with iptables [NET06]. All HTTP requests destined to port 80 are directed to port 3128, the service port of Squid. A layer 3 switch is acted as the ISP-side gateway. The bandwidth of the access link between the access gateway and the layer 3 switch is limited to 2Mbps. As the configuration in simulation, three classes are provided with service ratio 4:2:1. Notably, the cache function is disabled to avoid getting responses directly from caches. The effects of MSF-RS Squid are observed in terms of weighted fairness, user-perceived latency, and CPU loading as follows.

1) *Weighted fairness*: The amounts of bandwidth allocated to three classes for a 200-second test are 1.03 Mbps, 0.52 Mbps, and 0.26 Mbps, respectively, when backlogged requests are applied. The result quite obeys the configured service ratio 4:2:1.

2) *User-perceived latency*: Table 4.2 shows the latency provided by the original Squid and the MSF-RS Squid. The original-Squid case represents all requests are immediately released by the proxy. The MSF-RS Squid reduces $(1686-1175)/1686$, or 30%, of the average user-perceived latency in the original Squid case, although the user-perceived latency in MSF-RS includes the additional queuing time, 515.5 ms.

TABLE 4.2. USER-PERCEIVED LATENCY COMPARISONS

Item	Case	
	<i>Original Squid</i>	<i>MSF-RS Squid</i>
User-perceived latency	1686.1 ms	1174.9 ms (include queuing time 515.5 ms)

3) *CPU loading*: Table 4.3 shows the benchmark results on CPU time occupied by the MSF-RS Squid process and the original Squid process when both processes provide the same link-speed throughput during 200 seconds. As expected, the CPU time increases as the number of classes or the access link bandwidth increases. Notably, the time under MSF-RS is always lower than that under the original Squid. Under the original Squid, all requests are immediately released by the proxy, bringing great concurrent transactions. However, a proper number of concurrent transactions are allowed by MSF-RS. It is believed that the number of concurrent transactions dominates the cost of CPU computing.

TABLE 4.3. COMPARISON BETWEEN MSF-RS AND THE ORIGINAL SQUID ON CPU TIME

Case	Link Capacity	
	<i>2 Mbps</i>	<i>10 Mbps</i>
MSF-RS Squid with 10 Classes	44.8 secs	56.34 secs
MSF-RS Squid with 100 Classes	46.02 secs	58.04 secs
Original Squid	63.22 secs	84.90 secs

4.8 Summary

Scheduling the uplink requests is a potential method to manage the bottlenecked downlink at the *user-side* access gateway. Because the class-based fair queuing (FQ) discipline is widely and maturely used in scheduling packets, we first investigate the possibility of applying the discipline to schedule requests. However, we found that three problems occur at applying the class-based FQ discipline to schedule requests:

the *timing* and *ordering* to release requests and the suitability of class-based for user-level differentiation. Based on the investigation on the three problems, we propose the minimum-service first request scheduling (MSF-RS) scheme to manage the access link bandwidth at user-side access gateway. To achieve high bandwidth utilization while avoiding congesting the link, the window rate control module in MSF-RS determines the releasing rate of requests and the number of outstanding responses. To perform user-based weighted fairness and bandwidth sharing, the minimum-service arbiter module in MSF-RS always selects the request from the class receiving the least normalized responses.

The analysis proves that MSF-RS shortens 25% of the user-perceived latency on average, compared with an ordinary gateway, because the number of concurrent transmissions is controlled, even though this control may queue requests in the MSF-RS gateway. Besides, the existence of bounds on delay and fairness represents that the MSF-RS gateway does provide the differential service among classes while avoiding the low-class users from long latency. The results in the simulation and the field trial show that the bandwidth usage between classes conforms to the targeted ratio and the idle bandwidth is proportionally shared by all active classes. Besides, MSF-RS reduces 23.44% and 30% of user-perceived latency in the simulation and the field trial, respectively.

Chapter 5

Conclusions

This dissertation investigates and proposes fairness control schemes respectively to solve the public and private unfair problems which currently existing solutions cannot handle at the end host or at the user-side gateway. To maintain the public fairness in the Internet without the sacrifice of media flows on their deserved bandwidth, we promote that a good end-to-end rate control scheme should be TCP-equivalent, i.e. use the same throughput as TCP in the steady state, but as aggressive and responsive as TCP in the transient state. Moreover, it should be TCP equal-share, i.e. using bandwidth equal to TCP flows in the same bottleneck. On the other hand, to solve the private fairness problem which packet schedulers fail at the user-side gateway, this dissertation considers a solution of scheduling uplink requests to manage the downlink bandwidth.

We first exhibits that a TCP friendly scheme may have desirable TCP-equivalence and TCP equal-share to maintain public fairness, if it takes the rate-based fairness strategy, the historical/super-linear aggressiveness strategy, and the fixed history responsiveness strategy. Because no single existing scheme simultaneously takes the three recommended strategies to meet the public fairness, a window-averaging rate control (WARC) scheme is proposed based on the above observation and to be deployed in the end hosts. WARC averages the rate over a constant number of CWNDs, leading to its two inherent advantages: *fairness under stationary* conditions and *fast aggressiveness*. Besides, WARC employs the history-reset procedure to have *fast responsiveness* when the available bandwidth drops dramatically.

Next, we investigates the possibility of scheduling requests by using the class-based fair queuing discipline for the private fairness, then identify the difficulty on the timing and ordering determination of releasing requests and discuss the suitability of the class-based policy for user-level differentiation. By the investigation, a minimum-service first request-scheduling (MSF-RS) scheme is proposed and to be deployed in the user-side access gateway. MSF-RS always selects the request from the class receiving the least normalized responses to perform user-based weighted fairness and bandwidth sharing. Also, MSF-RS determines the releasing rate of

requests and the number of outstanding responses to achieve high bandwidth utilization while avoiding congesting the link.

The analysis and simulation results demonstrate that WARC does perform better TCP equivalence and TCP equal-share to maintain the public fairness than other TCP-friendly schemes, while WARC still provides the smoother rate than all of them. Besides, the analysis proves that MSF-RS has bounds on delay and fairness, representing that MSF-RS not only provides the differential service among classes for the private fairness but also avoids the low-class users from long latency, which are also demonstrated by the simulation and field trial. Besides, MSF-RS shortens 20%~30% of the user-perceived latency and 25% of CPU loading on average.



Appendix 1

Smoothness Level of TCP-friendly Schemes

The appendix displays the smooth level of the schemes to reveal (1) Except TFRCP, other schemes do have smoother throughput than TCP (<60% of CV of TCP throughput.) (2) Although the smooth levels among schemes are different, such differences do not affect our conclusions, that is, the suitable strategies and the recommendation for schemes.

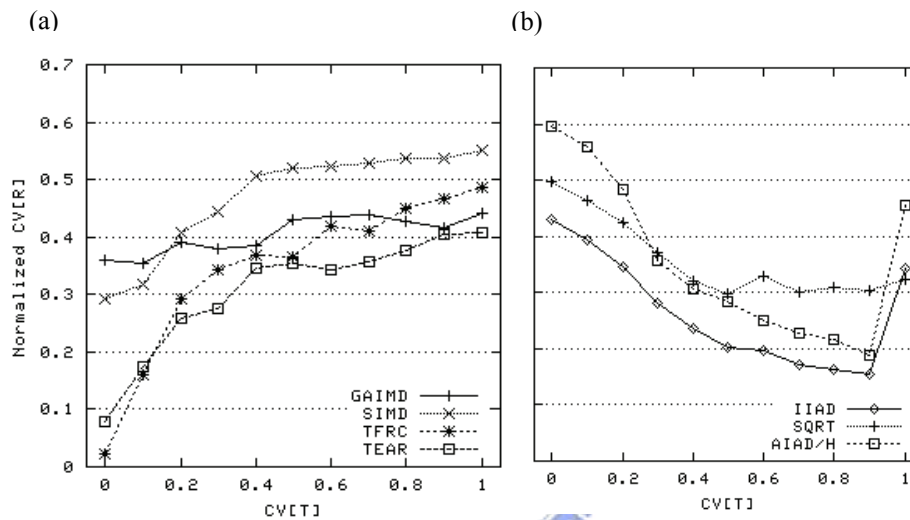


Fig. A1.1. The smooth level on the throughput of each scheme, relative to that of TCP, under different levels of variant-losses condition

For the non-periodic loss scenario (Chapter 2.4.1), we measure the CV of throughput for each scheme under different $CV[T]$, where T means the inter-loss time. Fig. A1.1 plots the CV of throughput, denoted as $CV[R]$, for each scheme, normalized with that of TCP.

TFRC and TEAR provide obviously smoother behavior under a less variant condition ($CV[T] < 0.5$). They postpone the rate increase until the time escaped from the last loss is longer enough. Such a delay-beginning mechanism thus provides them better smoothness feature than others when the inter-loss time is regular. IIAD, SQRT, and AIAD/H provide smoother throughput even when the loss condition is variant, but they sacrifice their deserved throughput, as shown in Fig. 2.1(b) and (c). IIAD and AIAD/H have the strange values as $CV[T]=1$, since occasionally their CWND may decrease to zero and then take long time to recover. TFRCP, unexpectedly, has worse $CV[R]$ than TCP under the testing, resulting from its *non-historical/super-linear*

increasing strategy. Its normalized $CV[R]$ is larger than one once $CV[T]>0.1$, and linearly increases to 2.2 as $CV[T]=1$.



Appendix 2

Analysis on WARC

Herein we describe the increase function of WARC and sequentially analyze its smoothness, aggressiveness and responsiveness.

A2.1 Increase Functions

The increase function of a scheme describes how the scheme increases its rate between two consecutive packet losses. The rate means the number of packets sent in one RTT. The increase functions of several well-known schemes, such as AIMD, SIMD, and AIAD, are expressed in [JGM03]. Fig. A2.1 plots the increase curve of TCP, GAIMD(1/5,1/8), and WARC.

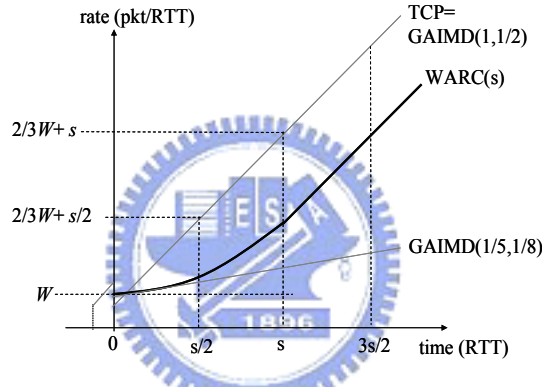


Fig. A2.1 The increase curves of WARC, GAIMD, and TCP

The increase function of WARC is calculated from that of TCP as follows. Assume that TCP stays at the steady state with an average rate W before the time 0, and then a packet loss occurs at the time 0. Because a TCP with an average rate W implies that its CWND increases from $2W/3$ to $4W/3$ and then falls back to $2W/3$ once after a packet loss, the CWND in the TCP at the time 0 is $2W/3$. Next, since TCP increases CWND one per RTT, the average CWND in TCP between the time $0 \sim t$ would be $2W/3 + t/2$. Recall that WARC gets the new rate by averaging the CWNDs of TCP over the latest s RTTs. Thus, before the time s ($t \leq s$), the rate of WARC can be averaged with weight from the mean rate before the time 0 and between the time 0 and t , as written by the expression

$$\begin{aligned} w(t) &= \frac{s-t}{s} \times W + \frac{t}{s} \times \left(\frac{2}{3}W + \frac{t}{2} \right) \\ &= W + \frac{t^2}{2s} - \frac{Wt}{3s}, \end{aligned} \quad (\text{A2.1})$$

where $w(t)$ represents the rate of WARC at time t (in RTTs) elapsed since the last loss and $w(0)=W$. After the time s ($t>s$), the rate of WARC equals to the average CWNDs of TCP between t and $t-s$, as written by the expression

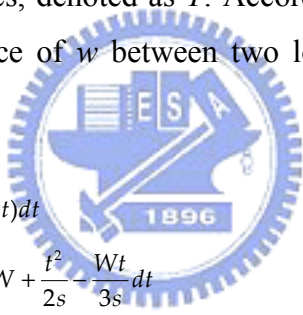
$$\begin{aligned} w(t) &= \frac{1}{2} \left(\left(\frac{2}{3}W + t - s \right) + \left(\frac{2}{3}W + t \right) \right) \\ &= \frac{2}{3}W + t - \frac{s}{2}. \end{aligned} \quad (\text{A2.2})$$

From (A2.1) and (A2.2), the increase function of WARC is expressed as

$$w(t) = \begin{cases} W + \frac{t^2}{2s} - \frac{Wt}{3s} & (t \leq s) \\ \frac{2}{3}W + t - \frac{s}{2} & (t > s) \end{cases}. \quad (\text{A2.3})$$

A2.2 Smoothness

The skill for the following analysis is derived from [JGM03]. Assume that x packets are sent between two losses by a flow with the mean rate W . Then, x/W will be the time between two losses, denoted as T . According to the increase function of WARC, the mean and variance of w between two losses, $E[w]$ and $Var[w]$ can be respectively expressed as



$$\begin{aligned} E[w] &= \frac{1}{T} \int_0^T w(t) dt \\ &= \begin{cases} \frac{1}{T} \int_0^T \left(W + \frac{t^2}{2s} - \frac{Wt}{3s} \right) dt & (T \leq s) \\ \frac{1}{T} \left(\int_0^s \left(W + \frac{t^2}{2s} - \frac{Wt}{3s} \right) dt + \int_s^T \left(\frac{2}{3}W + t - \frac{s}{2} \right) dt \right) & (T > s) \end{cases} \\ &= \begin{cases} \frac{1}{6s} T^2 - \frac{W}{6s} T + W & (T \leq s) \\ \frac{1}{2} T + \frac{(4W - 3s)}{6} + \frac{Ws + s^2}{6T} & (T > s) \end{cases} \end{aligned} \quad (\text{A2.4})$$

and

$$\begin{aligned} Var[w] &= \frac{1}{T} \int_0^T (w(t) - E[w(t)])^2 dt \\ &= \begin{cases} \frac{T^2}{540s^2} (12T^2 - 15WT + 5W^2) & (T \leq s) \\ \frac{1}{540T^2} \begin{pmatrix} 45T^4 - 15s^4 \\ +6s^3(12T - 5W) + 10sTW(-9T + 2w) \\ -15s^2(6T^2 - 7TW + W^2) \end{pmatrix} & (T > s). \end{cases} \end{aligned} \quad (\text{A2.5})$$

Herein we take the same assumption as that in [JGM03], x is i.i.d. exponential distributed with a mean $2W^2/3$. Next, we calculate the mean of $Var[w]$ over different loss probabilities as

$$\begin{aligned}
E[\text{Var}[w]] &= \int_0^\infty \text{Var}[w]P(T)dT \\
&= \int_0^\infty \text{Var}[w] \left(\frac{3}{2W^2} e^{-\frac{3}{2W^2}x} \right) dx \\
&= \int_0^{sW} \text{Var}_1[w] \left(\frac{3}{2W^2} e^{-\frac{3}{2W^2}x} \right) dx \\
&\quad + \int_{sW}^\infty \text{Var}_2[w] \left(\frac{3}{2W^2} e^{-\frac{3}{2W^2}x} \right) dx,
\end{aligned} \tag{A2.6}$$

where $\text{Var}_1[w]$ and $\text{Var}_2[w]$ are the two expressions shown in (A2.5), respectively. Finally, the smoothness of WARC can be expressed as

$$\begin{aligned}
\overline{\text{CV}}[w] &= \frac{\sqrt{E[\text{Var}[w]]}}{W} \\
&= \frac{\sqrt{\overline{\text{CV}}_1(s, W) + \overline{\text{CV}}_2(s, W)}}{W}
\end{aligned} \tag{A2.7}$$

where

$$\begin{cases} \overline{\text{CV}}_1(s, W) = \int_0^{sW} \text{Var}_1[w] \left(\frac{3}{2W^2} e^{-\frac{3}{2W^2}x} \right) dx \\ \overline{\text{CV}}_2(s, W) = \int_{sW}^\infty \text{Var}_2[w] \left(\frac{3}{2W^2} e^{-\frac{3}{2W^2}x} \right) dx \end{cases} \tag{A2.8}$$

Instead of showing the complex calculated result of (A2.8), we plot it in Fig. 3.5 to display the smoothness of WARC, comparing with SIMD and GAIMD.

A2.3 Aggressiveness

The aggressiveness, $\text{Aggr}(m)$, means the inverse of the time that a scheme increases its rate from $w(0)$ to $m \times w(0)$ [JGM03]. Suppose that WARC achieves the steady state before the time 0 and has the average rate W , i.e. $w(0)=W$ in (A1). Next, assume at the time T , WARC increases the rate to $m \times W$, i.e. $w(T)=m \times W$. Then, by solving T in (A2.1), we get $\text{Aggr}(m)$ of WARC, expressed as

$$\begin{aligned}
&\frac{1}{\text{Aggr}(m)} \\
&= T \\
&= \begin{cases} \frac{1}{3}(W + \sqrt{W^2 + 18msW - 18sW}) & (m \leq \frac{2}{3}W + \frac{s}{2W}) \\ (m - \frac{2}{3})W + \frac{1}{2}s & (m > \frac{2}{3}W + \frac{s}{2W}). \end{cases}
\end{aligned} \tag{A2.9}$$

A2.4 Responsiveness

The responsiveness, $\text{Resp}(m)$, means the inverse of the number of loss events required by a scheme to decrease the rate with a factor of $1/m$. Since the responsiveness of WARC would be fixed to N when the history-reset procedure is invoked, we analyze the responsiveness provided by the basic control of WARC, denoted as $\text{Resp}_{\text{basic}}(m)$. Assume that the inter-loss time is $m \times T$ RTTs before the

time 0, and it changes to T RTTs after the time 0. Suppose that WARC has a steady-state rate W before the time 0, where $W = 1.5m \times T$ which is the mean CWND that a TCP or TCP-equivalent flow should have under such an inter-loss time [ZDP01]. Fig. A2.2 plots the CWNDs of the potential TCP flow. The first start window w_1 of the TCP flow at the time 0 is $m \times T$. Then, its i^{th} start window w_i can be derived as

$$w_i = \frac{1}{2^{i-1}}(m-1)T + T.$$

Assume the \bar{w}_i denotes the mean CWNDs between the i^{th} and $(i+1)^{\text{th}}$ losses after the time 0, and then it can be expressed as

$$\begin{aligned} \bar{w}_i &= w_i + \frac{T}{2} \\ &= \frac{1}{2^{i-1}}(m-1)T + \frac{3}{2}T. \end{aligned}$$

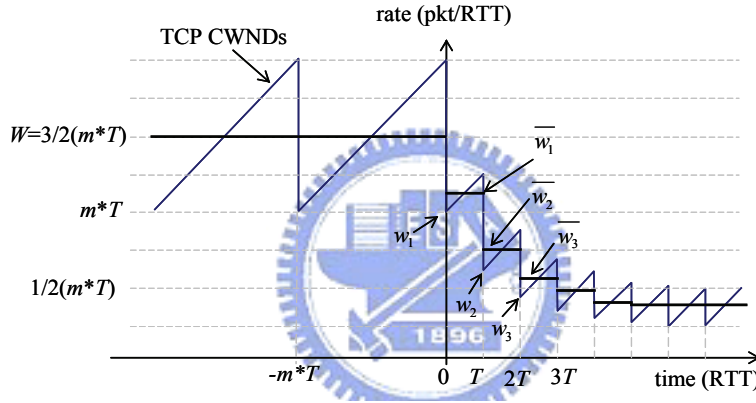


Fig. A2.2. The decreasing of the CWNDs and the mean CWNDs of a TCP flow when the inter-loss time change to T seconds after the time $m \times T$.

Since the packet rate in WARC is the average of the latest s CWNDs, the packet rate after n losses can be represented as

$$R(n) = \frac{1}{s} \left[(s-nT) \frac{3}{2}mT + T \left(\sum_{i=1}^n \bar{w}_i \right) \right], \quad (\text{A2.10})$$

where s is supposed larger than $n \times T$. Next, according to the definition of responsiveness, $1/\text{Resp}_{\text{basic}}(m)$ is the number of received losses before WARC reduces its rate, expressed by (A2.10), to $1.5T$. That is, $1/\text{Resp}_{\text{basic}}(m)$ equals to the n satisfying

$$R(n) \leq \frac{3}{2}T. \quad (\text{A2.11})$$

Finally, after T in (A2.10) and (A2.11) is replaced with $\frac{2W}{3m}$, by algebra we get that (A2.11) is true once if

$$n \geq \frac{4}{3} + \frac{3ms}{2W}.$$

Next, we consider under what m the history-reset procedure would be invoked. According to the description in Section 3.2, the HR procedure would be invoked if the present sending rate of WARC is larger than K times of $\bar{R}_{TCP}(N)$ where $\bar{R}_{TCP}(N) = 1.5T$ in this example based on (3.4) since $X(-j) = T$ ($1 \leq j \leq n$). That is, the HR procedure is invoked if the following equation is true.

$$R(N) > K \bar{R}_{TCP}(N) = \frac{3}{2} KT. \quad (\text{A2.12})$$

From (A2.12) and (A2.10) we can say that the HR procedure will be invoked if

$$\begin{aligned} m &> \frac{3 \cdot 2^N (Ks - N \cdot T) - 4T(1 + 2^N)}{3 \cdot 2^N (s - N \cdot T) - 4T(1 + 2^N)} \\ &\approx 1.1K. \end{aligned}$$



Appendix 3

Unfairness of TFRC and TEAR

Here we prove the unfairness of TFRC and TEAR under non-periodic loss model. A scheme following RTE represents that during the whole connection, it repeatedly estimates the TCP throughput and adjusts the packet rate to the estimated throughput. The long-term throughput of such a scheme can be expressed as

$$E[T_{RTE}] = \frac{\sum_{n=1}^{\infty} (R(n) \times L(n))}{\sum_{n=1}^{\infty} L(n)}, \quad (\text{A3.1})$$

where $L(n)$ is the length of the n^{th} rate-adjusting interval and $R(n)$ is the rate used in this interval.

A3.1 Unfairness of TFRC

The rate-adjusting interval L in TFRC is fixed. The packet rate in the n^{th} interval, R_n , is set as the mean rate a TCP flow had during the last time interval L_{n-1} . To estimate the mean rate, TFRC calculates the TCP throughput formula [PFT98], f , with the measured packet loss ratio p_{n-1} and RTT, where p_{n-1} is defined as the packet loss ratio in the last time interval L_{n-1} . The measurement on RTT is the same as that used in TCP.

Next, we show that in the long term the mean rate of TFRC, $E[T_{TFRC}]$, is equal to or larger than that of TCP, $E[T_{TCP}]$. Because L_n is fixed in TFRC, $E[T_{TFRC}]$ is a simplified form of Eq. (A3.1),

$$E[T_{TFRC}] = \lim_{v \rightarrow \infty} \frac{1}{v} \sum_{n=1}^v R(n) = E[R(n)]. \quad (\text{A3.2})$$

where $R(n)$ in TFRC is got by the TCP throughput formula f , p_{n-1} , and RTT. By assuming that RTT is fixed during the connection, Eq (A3.2) can be rewritten as

$$E[T_{TFRC}] = E[f(p_{n-1})] = E[f(p_n)]. \quad (\text{A3.3})$$

However, according to [PKT99],

$$E[T_{TCP}] = f(E[p_n]).$$

Thus,

$$E[T_{TFRCP}] = E[f(p_n)] \geq f[E[p_n]] = E[T_{TCP}]$$

because of the convexity⁶ of $f(p)$ [VB05].

TFRCP seems to assume that the loss condition measured in a fixed period represents the steady-state loss condition. That is, it expects that $p_n = E[p_n]$. The expression may be false particularly when p_n is varied. When the expression is false, the adjusted rate in TFRCP, $f(p_{n-1})$, will not equal to the mean rate of TCP averaged over a fixed period. However, WARC really averages the rate of TCP over a fixed period, so it can meet fairness even under the varied p_n .

A3.2 Unfairness of TEAR

The rate-adjusting interval L in the basic form of TEAR is the length of an epoch. Different from TFRCP, the rate adjusted in the present interval, $R(n)$, is not calculated with the TCP throughput formula. The $R(n)$ is averaged directly from the CWNDs of the TCP flow *emulated* at the receiver. By the emulation, TEAR gets the CWND of a TCP flow in each round and computes the mean CWND, \bar{W} , for each epoch at the end of each epoch. Next, TEAR sets $R(n)$ to a weighted average of \bar{W} , that is

$$R(n) = \sum_{j=1}^8 c_j \bar{W}_{n-j} \quad (\text{A3.4})$$

where $\{c_j\}$ is a series of weights with sum equal to 1.

Because the mean window in an epoch is the weighted average of the historical inter-loss time, i.e. $\bar{W}_n = \sum_{i=1}^8 u_i L_{n-i}$, Eq. (A3.4) can be rewritten with L directly, shown

as

$$R(n) = \sum_{i=1}^{16} c'_i L_{n-i} \quad (\text{A3.5})$$

⁶ A function $f(x)$ is convex if for any two point x_1 and x_2 , $f[\frac{1}{2}(x_1 + x_2)] \leq \frac{1}{2}[f(x_1) + f(x_2)]$.

where $\{c_j\}$ is a series of production with c_i and u_i and $\sum_{i=1}^{16} c'_i = \frac{3}{2}$ since $\sum_{i=1}^8 u_i = \frac{3}{2}$.

Next, we show that in the long term the mean rate of TEAR, $E[T_{\text{TEAR}}]$, is equal or smaller than that of TCP. By rewriting Eq. (A3.1) with Eq. (A3.5) and assuming that the distribution of the inter-loss time is i.i.d., we can get

$$\begin{aligned}
 E[T_{\text{TEAR}}] &= \frac{\sum_{n=0}^{\infty} \left(\left(\sum_{i=1}^{16} c'_i L_{n-i} \right) L_n \right)}{\sum_{n=0}^{\infty} L_n} = \sum_{i=1}^{16} c'_i \frac{\sum_{n=0}^{\infty} (L_n)^2}{\sum_{n=0}^{\infty} L_n} \\
 &= \frac{3 E[L_n^2]}{2 E[L_n]} \tag{A3.6} \\
 &\leq \frac{3 E[L_n]^2}{2 E[L_n]} = \frac{3}{2} E[L_n] = E[T_{\text{TCP}}].
 \end{aligned}$$

According to Eq. (A3.6), we conclude that $E[T_{\text{TEAR}}]$ would equal to $E[T_{\text{TCP}}]$ only if

$E[L_n^2] = E[L_n]^2$, that is the inter-loss time is fixed, or the loss arrives periodically.

Otherwise, $E[T_{\text{TEAR}}]$ will smaller than $E[T_{\text{TCP}}]$.



References

- [AAB05] E. Altman, K. Avrachenkov, and C. Barakat, "A Stochastic Model of TCP/IP with Stationary Random Losses," *IEEE/ACM Transactions on Networking*, vol. 13, no 2, pp. 356-369, April 2005.
- [AVA06] Avalanche, http://www.spirentcom.com/analysis/product_line.cfm?PL=32.
- [BB01] D. Bansal and H. Balakrishnan., "Binomial Congestion Control Algorithms," In *Proc. of IEEE INFOCOM*, April 2001, pp. 631-640.
- [BBF01] D. Bansal, H. Balakrishnan, S. Floyd, S. Shenker, "Dynamic Behavior of Slowly-responsive Congestion Control Algorithms," in *Proc. of ACM SIGCOMM'01*, Aug. 2001, pp. 263-274.
- [BBK00] N. Bhatti, A. Bouch, and A. Kuchinsky, "Integrating User-Perceived Quality into Web Server Design," Proceedings of the 9th International World Wide Web Conference, 2000.
- [BC98] P. Barford and M. Crovella, "Generating Representative Web Workloads for Network and Server Performance Evaluation," *ACM SIGMETRICS Performance Evaluation Review*, vol. 26, issue 1, pp. 151-160, June 1998.
- [BCC98] B. Braden, D. Clark and J. Crowcroft, "Recommendations on Queue Management and Congestion Avoidance in the Internet," RFC 2309, Apr 1998, Available at <http://www.ietf.org>.
- [BP95] L. Brakmo and L. Peterson, "TCP Vegas: End to End Congestion Avoidance on a Global Internet," *IEEE Journal on Selected Areas in Communication*, vol 13, no. 8, Oct. 1995, pp. 1465-1480.
- [CC01] E. Casalicchio and M. Colajanni, "A Client-Aware Dispatching Algorithm for Web Clusters Providing Multiple Services," Proceedings of the 10th International World Wide Web Conference, 2001.
- [CCC02] V. Cardellini, E. Casalicchio, M. Colajanni, and M. Mambelli, "Enhancing a Web-Server Cluster with Quality of Service Mechanisms," Proceedings of IEEE International Performance Computing and Communications Conference, 2002.
- [CKD02] M. Conti, M. Kumar, S. K. Das, and B. A. Shirazi, "Quality of Service Issues in Internet Web Services," *IEEE Trans. on Computers*, vol. 51, no. 6, June 2002.
- [CLB04] Gwyn Chatrnon, Miguel Labrador, and Sujata Banerjee, "Fairness of AQM Schemes for TCP-friendly Traffic" in *Proc. of IEEE Globecom*, Dallas, Dec. 2004, pp 721-731.
- [CP99] L. Cherkasova and P. Phaa, "Session Based Admission Control: a Mechanism for Web QoS," Proceedings of the International Workshop on Quality of Service, 1999.
- [F00] S. Floyd, "Congestion Control Principles," RFC 2914, Sept. 2000.
- [FF99] Floyd, S., and Fall, K., "Promoting the Use of End-to-End Congestion Control in the Internet." *IEEE/ACM Transactions on Networking*, August 1999.
- [FHP00] S. Floyd, M. Handley, J. Padhye, and J. Widmer, "Equation-based Congestion Control for Unicast Applications," in *Proc. of ACM SIGCOMM'00*, Aug 2000, pp. 43-56.

- [FJ95] S. Floyd, and V. Jacobson, "Link-sharing and Resource Management Models for Packet Networks," *IEEE/ACM Transactions on Networking*, Vol. 3 No. 4, pp. 365-386, August 1995.
- [G04] S. Gorinsky, "Feedback Modeling in Internet Congestion Control", in *Proc. of Next Generation Teletraffic and Wired/Wireless Advanced Networking (NEW2AN 2004)*, Feb. 2004, pp. 231-234.
- [GOL94] J. Golestani, "A Self-Clocked Fair Queueing Scheme for Broadband Applications," *Proceedings of the IEEE INFOCOM*, Toronto, June 1994.
- [GTC06] L. Guo, E. Tan, S. Chen, Z. Xiao, O. Spatscheck and X. Zhang, "Delving into Internet Streaming Media Delivery: A Quality and Resource Utilization Perspective," in *Proc. of ACM IMC'06*, Oct. 2006.
- [GVC96] P. Goyal, H. Vin, and H. Chen, "Start-Time Fair Queueing: A Scheduling Algorithm for Integrated Services Packet Switching Networks," *Proceedings of the ACM SIGCOMM*, August 1996.
- [JGM03] Shudong Jin, Liang Guo, Ibrahim Matta, and Azer Bestavros, "A Spectrum of TCP-friendly Window-based Congestion Control Algorithms," *IEEE/ACM Transactions on Networking*, vol. 11, no. 3, pp. 341-355, June 2003.
- [JID04] S. Jaiswal, G. Iannaccone, C. Diot, J. Kurose, Towsley, D., "Inferring TCP connection characteristics through passive measurements," *IEEE INFOCOM 2004*, Mar. 2004, pp. 1582-1592.
- [JLH07] D. Wei, C. Jin, S. H. Low, and S. Hegde, "FAST TCP: Motivation, Architecture, Algorithms, Performance," *IEEE/ACM Transactions on Networking*, vol. 14, no. 6, pp. 1246-1259, 2007.
- [KHF06] E. Kohler, M. Handley, and S. Floyd, "Designing DCCP: congestion control without reliability," *ACM SIGCOMM Computer Communication Review*, vol. 36, no. 4, Sept. 2006.
- [KHR02] D. Katabi, M. Handley, and C. Rohrs, "Congestion Control for High Bandwidth-Delay Product Networks," in *Proc. of ACM SIGCOMM'02*, Aug. 2002, pp. 89-102.
- [LG07] Ladan Gharai, "RTP with TCP Friendly Rate Control," draft-ietf-avt-tfrc-profile-10.txt, IETF Internet-draft, 22 July, 2007.
- [LGC01] C. Li, G. Peng, K. Gopalan, and T. Chiuch, "Performance Guarantee for Cluster-Based Internet Services," State University of Stony Brook, May 2001.
- [LM07] Live Network, Inc. "LIVE555 Streaming Media," <http://www.live555.com/liveMedia/>
- [LPW03] S. H. Low, F. Paganini, J. Wang, and J. Doyle, "Linear Stability of TCP/RED and a Scalable Control," *Computer Networks Journal*, vol. 43, no. 5, pp. 633-647, Dec. 2003.
- [LT03] Lahanas A and Tsaoussidis V, "Exploiting the efficiency and fairness potential of AIMD-based congestion avoidance and control," *Computer Networks*, vol 43, no 2, pp. 227-245, Oct. 7 2003.
- [MMF96] M. Mathis, J. Mahdavi, S. Floyd, and A. Romanow, "TCP Selective Acknowledgment

- Options, “ RFC 2018, April 1996.
- [NFT06] The netfilter/iptables project, <http://www.netfilter.org/>.
- [NK04] N. Itaya and S. Kasahara, “Dynamic Parameter Adjustment for Available-bandwidth Estimation of TCP in Wired-wireless Networks,” *Computer Communications*, vol. 27 no. 10, pp. 976-988, June 2004.
- [NS06] The Network Simulator - ns-2, 2.1b9, <http://www.isi.edu/nsnam/ns/>.
- [PBB98] R. Pandey, J. Fritz Barnes, and R. Fritz Barnes, “Supporting Quality of Service in HTTP Servers,” Proceedings of the Seventeenth Annual ACM Symposium on Principles of Distributed Computing, pp. 247-256, 1998.
- [PFT98] J. Padhye, V. Firoiu, D. Towsley, and J. Kurose, “Modeling TCP throughput: A simple model and its Empirical Validation,” in *Proc. of ACM SIGCOMM’98*, Sep. 1998, pp. 303-314.
- [PG93] A. K. Parekh and R. G. Gallager, “A Generalized Processor Sharing Approach to Flow Control in Integrated Services Networks: The Single-Node case,” *IEEE/ACM Trans. on Networking*, pp. 344-357, June 1993.
- [PG93] A. K. Parekh and R. G. Gallager, “A Generalized Processor Sharing Approach to Flow Control in Integrated Services Networks: The Single-Node case,” *IEEE/ACM Trans. on Networking*, pp. 344-357, June 1993.
- [PKT99] J. Padhye, J. Kurose, D. Towsley, and R. Koodli, “A Model Based TCP-friendly Rate Control Protocol,” In *Proc. of NOSSDAV*, June 1999.
- [RHE99] R. Rejaie, M. Handley, and D. Estrin, “Rap: An End-to-end Rate-based Congestion Control Mechanism for Real-time Streams in the Internet,” In *Proc. of IEEE INFOCOM*, Mar. 1999, pp. 1337-1345.
- [RKL01] Y. R. Yang, M. S. Kim, and S. S. Lam, “Transient behavior of TCP friendly congestion control protocols,” in *Proc. of IEEE INFOCOM*, Apr. 2001, pp. 1716-1725.
- [ROY00] I. Rhee, V. Ozdemir, and Y. Yi, “TEAR: TCP Emulation at Receivers-Flow Control for Multimedia Streaming,” Tech. Rep., Department of Computer Science, NCSU, Apr. 2000.
- [SCF03] H. Schulzrinne, S. Casner, R. Frederick, and V. Jacobson, “RTP: A Transport Protocol for Real-Time Applications,” RFC 3550.
- [SQI06] Squid Web Proxy Cache, <http://www.squid-cache.org/>.
- [SV96] M. Shreedhar and G. Varghese, “Efficient Fair Queuing Using Deficit Round-Robin,” *IEEE/ACM transactions on networking* vol. 4, no. 3, June 1996.
- [SW00] D. Sisalem and A. Wolisz, “Lda+ tcp-friendly adaptation: A measurement and comparison study,” in *Proc. of NOSSDAV*, June 2000.
- [TL01] Shih-Chiang Tsao and Ying-Dar Lin, “Pre-order Deficit Round Robin: A New Scheduling Algorithm for Packet-switched Networks,” *Computer Networks*, Vol. 35(2-3), pp. 287-305, 2001.
- [TWH05] K. A. Tang, J. Wang, S. Hegde, and S. H. Low, “Equilibrium and Fairness of Networks

Shared by TCP Reno and FAST”, Telecommunications Systems special issue on High Speed Transport Protocols, vol. 30, no. 4, pp. 417-439, Dec. 2005.

- [TZ05] V. Tsaoussidis and C. Zhang, “The Dynamics of Responsiveness and Smoothness in Heterogeneous Networks,” *IEEE J SEL AREA COMM*, vol. 23, no. 6, pp. 1178-1189, June 2005.
- [VB05] Milan Vojnovic and J.-Y. Le Boudec, "On the Long-Run Behavior of Equation-Based Rate Control," *IEEE/ACM Transactions on Networking*, vol. 13, no 3, pp. 568-581, June 2005.
- [WDM01] J. Widmer, R. Denda and M. Mauve, “A survey on TCP-friendly congestion control,” *Special Issue of the IEEE Network `Control of Best Effort Traffic`*, vol. 15, no 3, pp. 28-37, May/June 2001.
- [WTL04] H. Y. Wei, S. C. Tsao, and Y. D. Lin, “Assessing and Improving TCP Rate Shaping Over Edge Gateways,” *IEEE Trans. on Computer*, vol. 53, issue 3, pp. 259-275, March 2004.
- [XSS05] Y. Xia, L. Subramanian, I. Stoica, and S. Kalyanaraman, "One More Bit Is Enough", in *Proc. of ACM SIGCOMM'05*, Aug. 2005
- [YL00] Y. Yang and S. Lam, “General AIMD Congestion Control,” in *Proc. of IEEE ICNP 2000*, Nov 2000, pp. 187-198.
- [ZDP01] Y. Zhang, N. Duffield, V. Paxson, and S. Shenker, “On the Constancy of Internet Path Properties,” in *Proc. of ACM SIGCOMM Internet Measurement Workshop*, Nov. 2001, pp 197-211.
- [ZT06] C. Zhang and V. Tsaoussidis, "TCP Smoothness and Window Adjustment Strategy", *IEEE Transactions on Multimedia*, vol 8, no. 3, pp. 600-609, June 2006.