Available online at www.sciencedirect.com

SCIENCE DIRECT®

Computer Communications 28 (2005) 108–124

computer communications

www.elsevier.com/locate/comcom

Lifetime-based TCP service differentiation

I. Nikolaidis*,[1], X. Wu

Computing Science Department, University of Alberta, Edmonton, Alta., Canada T6G 2E8

Received 18 June 2004; accepted 18 June 2004
Available online 19 August 2004

Abstract

We present a lifetime-based differentiation framework for TCP flows. The separation into two classes is based on a threshold technique. We introduce a scheme, `FairShare`, that handles the long-lived flows and achieves global max–min fairness. The short-lived flows are bundled together and a separate family of mechanisms, `DAS`, dynamically allocate bandwidth to match the load of newly instantiated short flows. Thus, two different objectives are met: fairness for the long flows, as well as reduced response times and reduced response time variance for the short flows. We argue that the applications are better served this way. Namely, applications generating short transfers are predominantly interested in short response times (e.g. HTTP requests/responses) while those generating long transfers (e.g. long FTP transfers) are at least provided a guarantee they are not penalized compared to other similar connections. By way of an example, we also demonstrate that elaborate traffic control schemes that do not perform classification of flows based on their anticipated lifetimes, may fail to efficiently utilize the network links.

Keywords: TCP; Fairness; Bandwidth allocation; Active queue management

1. Introduction

Research on the topic of controlling networks carrying TCP traffic, such as the Internet, frequently confronts the question whether information about ‘flows’, i.e. the *call*-level dynamics of the system, should be collected and used in controlling the network. In telephone networks, call-level control is certainly the norm. In the Internet, a flow/call is recognized as such only by the endpoints and not by the network interior, i.e. not by the routers on the path from source to destination. A quick review of the relevant literature suggests that the lifetime of TCP flows follows a heavy-tailed distribution [1,6,13]. That is, TCP flows are in their majority short-lived but a small fraction of long-lived flows accounts for a large fraction of the total carried traffic. It is questionable whether dealing with each short TCP flow individually for resource allocation makes any sense. The volume of control plane signalling would be prohibitive, and even then, the horizon over which it can assist the resource allocation process is limited due to the quick termination of such connections. Therefore, it is preferable to deal with short flows in one *bulk* class, investing on state information for the entire class instead for individual flows. The particular approach admits as a possible design one where the *long* flows are *still* treated on a per-flow basis. The heavy-tailed behavior of connection lifetimes also suggests that, at any point in time, only a handful of flows crossing a link are long-lived, thus, the total state overhead necessary to keep track of the long flows appears to be manageable.

The view taken in this paper is that long and short flows should be treated separately in terms of received service. Towards this end, we use a threshold-based classification of flows into short and long. However, the most important element of the presented work is that the separation of flows according to their lifetime is not only beneficial in terms of state overhead, but also beneficial for the performance received by end users. Namely, we conjecture that the most significant performance attribute for a short flow is its

* Corresponding author.
E-mail addresses: yannis@cs.ualberta.ca (I. Nikolaidis), xudong@cs.ualberta.ca (X. Wu).

[1] Supported in part by an operating grant from the Natural Sciences and Engineering Research Council of Canada (NSERC).

0140-3664/$ - see front matter © 2004 Elsevier B.V. All rights reserved.
doi:10.1016/j.comcom.2004.06.015

response time[2] while *fairness* is important (and better defined) for long-lived flows. Two observations support this particular thesis. First, the bulk of short flows are the results of HTTP request/responses [6]. Clearly, users are keen on receiving responses without any undue delay. Furthermore, avoiding frequent large response times (thus avoiding large variance of the response times) enhances the user's perceived performance of the network as being more consistent. The second element relates to long flows that are invariably long file transfers, over FTP or HTTP, and streaming of data. The *least* guarantee that should be provided to users is that their long flow is not treated any worse than any other long flow. Thus, fairness appears to be natural and essential for long flows. Moreover, it is technically difficult, and even non-sensical, to consider fairness between a short TCP flow and a long TCP flow. The short flows are inherently limited by the initial transient of the TCP window adjustment, being by their very design at a disadvantage against long flows. On the other hand, long flows amount for a large fraction of the packets transferred and controlling their long term behavior is one of the ways that the entire network performance can be controlled.

Lifetime-based classification schemes were originally presented in Refs. [9,14,16,21,24] to protect short TCP flows from the negative impact of the long ones. In particular, short flows are at a disadvantage when competing against long ones. This is due to the conservative nature of TCP congestion control: short flows usually operate with small congestion window size in the exponential growth phase. Since short flows have less data to transfer, they terminate within a few round-trip times (RTTs) without enough time to enlarge their congestion window to enter the congestion avoidance phase. The operation with relatively small congestion window impacts the short flows in two ways. It imposes a limit on the delivery rate of packets, but also renders connections more fragile to packet loss. Since long flows operate in congestion avoidance, upon packet loss they usually reduce their delivery rate less drastically, by reducing their congestion windows by half. However, packet losses are likely to initiate timeouts in short flows because few packets are in transit to allow for the triple-dupACK 'fast retransmission' to be invoked. Another disadvantage of short-lived TCP flows is that adequate RTT samples are unavailable, and hence the retransmission timeout (RTO) value is usually a large (conservative) value possibly equal to the estimated initial timeout value of RTO, which is much larger than typically observed RTT values in the Internet.

Summarizing, there appears to exist no particular advantage to mixing short- and long-lived flows together. Indeed, there appear to exist evidence in support of separation. The only compelling reason for bundling short and long flows together is for the benefit of multiplexing. Nevertheless, we can still achieve multiplexing without losing sight of the fact that the performance needs of the two classes can be met using (flexible) boundaries between them. Such boundaries can be enforced in the form of controlled (scheduled) multiplexing. We are therefore proposing a hybrid DiffServ and IntServ paradigm with (a top-level) DiffServ across lifetime classes, and IntServ applied within only the long-lived flow class. In the familiar abstraction of routing domains, we expect the classification of flows into short and long to take place at the access or edge routers, presenting to the core routers one class for the bulk of all short flows and one class, with individually accounted for, long flows. The two classes could even be considered as routed, in principle, independently of each other. We note that the design presented here is meant to provide a least common denominator to separate and service accordingly short and long flows. Further refinement of the classification, with possibly separate classes for UDP or specific applications can be built upon the basic design. Suffice is to say that once separated into classes, a minimum requirement is to allocate per-class bandwidth at the link schedulers in the core of the network. Such allocation can be accomplished by one of the many variants of weighted fair queueing (WFQ) that are increasingly commonplace in modern router designs.

The rest of the paper is divided into two major parts. Part I, describes how a single loss scheduler for long flows can be used to drive the flows sharing a link to max–min fairness, and in the process drive the entire network of long flows into global fairness. The details of the scheme, called `FairShare` are presented in Section 2. Part II, describes how the short versus long flow separation can be performed and how following the dynamics of the number of short flows admitted into the system can be used to guide the bandwidth allocated to the short flow class, leading to improved response times for the short flows and without sacrificing any of the properties attained by `FairShare` for the long flows. The details of the dynamic allocation of bandwidth to short flows are given in Section 3. Finally, Section 4 summarizes the results produced so far, places our work within the context of previous related research, and reviews certain technical issues that need further refinement. Throughout this paper we use two 'adversaries' when comparing the performance of our schemes. These are DropTail, representing simplicity, and random early detection (RED) [2,3] representing a scheme that does not compromise simplicity while attaining higher flexibility.

2. Part I: long TCP flows

We first deal with the issue of fairness among long-lived flows. TCP unfairness, rooted in the different RTT values of different TCP flows, has been identified a decade ago, but

[2] We define as *response time* the duration between the timepoint when a connection setup starts and until the timepoint when the last data packet of the connection is successfully acknowledged.