Available online at www.sciencedirect.com

SCIENCE DIRECT®

Computer Communications 29 (2006) 801–811

computer communications

www.elsevier.com/locate/comcom

Edge-to-edge proactive congestion control for aggregated traffic[☆]

Sergio Herrería-Alonso[*], Manuel Fernández-Veiga, Miguel Rodríguez-Pérez, Andrés Suárez-González, Cándido López-García

Department of Telematics Engineering, University of Vigo, ETSE Telecomunicación, Campus universitario s/n, 36310 Vigo, Spain

Available online 9 September 2005

Abstract

We present *Ping Trunking*, a novel edge-to-edge management technique that can provide soft service guarantees to aggregate traffic streams without requiring any special support at the core of the network. Our proposal is designed to work over aggregated flows that bundle a varying number of user flows for common treatment between two nodes in a network. To regulate the user data transmission rate, a Vegas-like management connection is established between the two edges of each aggregate. This control connection injects control packets into the network to probe its congestion level. Thanks to this managing, *Ping Trunking* is able to fairly share the network bandwidth among competing aggregates in accordance with their subscribed target rates. In addition, it does not cause undesired sharp variations in the transmission rates of handled aggregates and avoids packet losses at the core nodes. We demonstrate analytically and through simulation experiments the effectiveness of our technique.
© 2005 Elsevier B.V. All rights reserved.

Keywords: Aggregated traffic; Control overlay; Congestion control; TCP Vegas

1. Introduction

Only two simple key principles, namely the best effort paradigm for data transport and the end-to-end philosophy for traffic control and management, have prevented the current Internet to collapse while it underwent an exponential growth. However, the best effort model with no service guarantee is no longer acceptable in view of the proliferation of interactive applications such as Internet telephony or video conferencing. Along the past years, the IETF has standardized several frameworks to meet the demand for Quality of Service (QoS) support such as RSVP [2] in the control plane or Differentiated Services [3] in the architecture area.

For scalability reasons, it is impractical to enforce performance guarantees at a fine-grained level (e.g. per flow) and so the QoS requirements should be applied to aggregate traffic streams rather than to individual flows. An aggregate traffic stream bundles a number of flows for common treatment between two nodes in a network. Such form of aggregation clearly simplifies the allocation of network resources and promotes the deployment of QoS frameworks notably. For similar reasons, service providers are likely to offer performance commitments on a per-aggregate basis, either to end users or to other peer providers. Within this context, and irrespective of the end-to-end transport-level protocols used by individual flows, it becomes essential that aggregated traffic is responsive to network congestion at an aggregate level.

Several approaches proposed to apply congestion control to aggregates such as *Congestion Manager* [4] and *Coordination Protocol* [5] architectures require the modification of user applications at the endpoints and, therefore, they are not likely to be employed in the Internet backbone. Alternatively, the *TCP Trunking* technique [6,7] can be an excellent applicant to manage aggregate traffic streams in backbone networks because it can carry out this regulation without changing neither user protocols, nor applications. *TCP Trunking* enforces flow control for the customer aggregates at their ingress nodes, and in order to regulate the flow of each aggregate traffic stream into the network, it employs various control TCP connections that inject control packets into the network to probe its current congestion level.

In this paper, we introduce *Ping Trunking*, an enhancement of the *TCP Trunking* technique that improves the original one by changing the control overlay. In particular,

[☆] A shorter version of this paper appears in [1].
[*] Corresponding author. Tel.: +34 986 813901; fax: +34 986 812116.
E-mail address: sha@det.uvigo.es (S. Herrería-Alonso).

0140-3664/$ - see front matter © 2005 Elsevier B.V. All rights reserved.
doi:10.1016/j.comcom.2005.08.002

instead of using several TCP connections to manage each aggregate, a unique preventive Vegas-like connection is used. Due to the new control overlay, our proposal presents some advantages over *TCP Trunking*. First, it operates the management of aggregates more efficiently since only one control connection per aggregate is required. Second, it does not cause sharp variations in the transmission rate of the trunks and also reduces the size of the queues at the core of the network, due to the dynamics of its Vegas-like response to congestion. Third, it renders the achieved bandwidth for an aggregate insensitive to its round-trip time (RTT). In addition, since it does not use packet losses as congestion signal, the stability in high-speed networks is increased. Finally, it also introduces far less overhead than the original approach leaving more room for user data.

The remainder of this paper is organized as follows. In Section 2, we give a brief overview of *TCP Trunking*, the traffic management technique which *Ping Trunking* is based on. Section 3 describes the *Ping Trunking* mechanism. Section 4 contains some simulation experiments that validate the proposed design. *TCP* and *Ping Trunking* techniques are compared in Section 5. Section 6 analyzes the persistent congestion problem found in Vegas operation and its effects on our framework. In Section 7, we discuss various issues related to the deployment of our proposal. Section 8 describes other approaches proposed to manage aggregated traffic. We end the paper with some concluding remarks and future lines in Section 9.

2. TCP Trunking

A TCP trunk [6,7] is an aggregate traffic stream where data packets are transmitted at a rate dynamically determined by the TCP congestion control algorithm [8]. Each trunk carries a varying number of user flows between two nodes of the network. The flow of the aggregated traffic is regulated by a control TCP connection established between the two edges of the trunk. This control TCP connection injects control TCP packets into the network to probe its congestion level.

TCP Trunking fully decouples control from data transmission. The introduction of control packets is not conditioned by user data protocols, but it is based on control packet drops as in usual TCP connections. In addition, a TCP trunk will not retransmit user packets if they are lost. If it is required, retransmissions should be handled by the user applications on the end hosts.[1]

TCP Trunking is implemented in the following way. User packets arriving at ingress nodes are temporarily queued into the buffer of their corresponding trunk. After a control packet is transmitted, user packets can be forwarded, totalling at most virtual maximum segment size (*vmss*) bytes. When *vmss* user bytes have been transmitted, the control connection will generate and send a control packet if its control TCP congestion window allows it. This way, control packets regulate the transmission of user packets and then, the loss of control packets in the network not only slows down the transmission rate of control packets but also reduces the transmission rate of user data.

To smooth bandwidth variations, multiple control TCP connections are employed with each trunk. If each trunk were regulated by a single control connection, a control packet loss would cause the entire trunk to halve its sending rate, as dictated by the TCP dynamics. When dealing with aggregated traffic, such abrupt reduction in transmission rate on packet losses is undesirable. In [7], the authors argue that four control connections per trunk are enough to produce smooth bandwidth transitions.

To conclude with this overview, it is important to point out that both user and control packets must follow the same path between the edges of the trunk to ensure that control connections are probing the proper available bandwidth. This assumption can be absolutely guaranteed if TCP trunks are run on top of ATM virtual circuits or MPLS label-switched paths [9].

3. Ping Trunking

Our proposal, named *Ping Trunking*, borrows from *TCP Trunking* the key concept of decoupling control from data in IP networks, but the original technique has been ameliorated by changing the control overlay. *Ping Trunking* only establishes a single control connection between the two network edges of the trunk. This connection controls the flow of user packets into the core of the network using a Vegas-like congestion control mechanism that adapts its congestion window (*cwnd*) based on the measured changes in the RTT, and not only on the loss of control packets.

Fig. 1 provides finer detail on the operation of this mechanism. Incoming user packets at ingress nodes are classified as belonging to a particular trunk and queued in

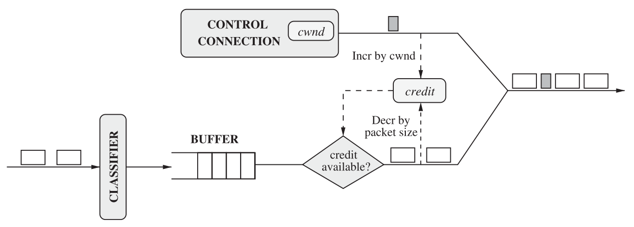

Fig. 1. *Ping Trunking* block diagram. This figure includes several simplifications. In fact, each trunk has its own buffer, credit bucket and control connection.

[1] The use of control packets in *TCP Trunking* is thus similar to that of resource management cells in ATM virtual circuits.