Discrete Applied Mathematics 156 (2008) 3187–3193

Contents lists available at ScienceDirect

Discrete Applied Mathematics

journal homepage: www.elsevier.com/locate/dam

Counting preimages of TCP reordering patterns

Anders Hansson [a], Gabriel Istrate [b,*]

[a] Information Sciences (CCS-3), Los Alamos National Laboratory, P.O. Box 1663, Mail Stop M997, Los Alamos, NM 87545, USA

[b] eAustria Research Institute, Bd. V. Pârvan no. 4, cam. 045B, Timişoara, RO-300223, Romania

ARTICLE INFO

Article history:
Received 15 October 2005
Received in revised form 20 July 2007
Accepted 21 May 2008
Available online 30 June 2008

Keywords:
TCP
Packet reordering
Doubly convex bipartite graphs
Matchings

ABSTRACT

Packet reordering is an important property of network traffic that should be captured by analytical models of the Transmission Control Protocol (TCP). We study a combinatorial problem motivated by RESTORED [G. Istrate, A. Hansson, S. Thulasidasan, M. Marathe, C. Barrett, Semantic compression of TCP traces, in: F. Boavida (Ed.), Proceedings of the Fifth IFIP NETWORKING Conference, in: Lecture Notes in Computer Science, vol. 3976, Springer-Verlag, 2006, pp. 123–135], a TCP modeling methodology that incorporates information about packet dynamics. A significant component of this model is a many-to-one mapping B that transforms sequences of packet IDs into *buffer sequences* in a manner that is compatible with TCP semantics. We obtain the following results:

- We give an easy necessary and sufficient condition for an input sequence W to be *valid* (i.e. $A \in B^{-1}(W)$ for some permutation A of $\{1, 2, \ldots, n\}$), and a linear time algorithm that, given a valid buffer sequence W of length n, constructs a permutation A in the preimage of W.
- We show that the problem of counting the number of permutations in $B^{-1}(W)$ has a polynomial time algorithm.
- We also show how to extend these results to sequences of IDs that contain repeated packets.

1. Introduction

Consider a sequence of TCP *packets*, identified by their integer IDs, as handled by their *receiver*. The receiver must forward the packet sequence to an *application*, subject to respecting *packet sequence integrity*. That is, at every moment the IDs of packets forwarded to the application must form a contiguous sequence $1, 2, \ldots, m$, for some $m \geq 1$. Packets can arrive out of order and thus need to be buffered. Several copies of a packet can arrive, but only one copy of a given packet is useful (and will be stored, if needed). We assume that the receiver evicts a given packet from the buffer and passes it to the application as soon as possible, i.e., as soon as the packet sequence integrity constraint is satisfied.

A given sequence $A = (A_1, \ldots, A_n)$ of packet IDs yields a corresponding sequence $B(A) = (B_{A,1}, \ldots, B_{A,n})$ representing the evolution of the buffer size. In this paper we are interested in the following problem: given a sequence of positive integers W, what is the complexity of:

(1) Deciding whether there exists a permutation A with $W = B(A)$?
(2) Counting the number of permutations in the set $B^{-1}(W)$?

We will give linear (respectively polynomial) time algorithms for these problems.

* Corresponding author. Fax: +40 256 244834.
E-mail addresses: hansson@lanl.gov (A. Hansson), gabrielistrate@acm.org (G. Istrate).

0166-218X/$ – see front matter © 2008 Elsevier B.V. All rights reserved.
doi:10.1016/j.dam.2008.05.011

2. Motivation

The problem we described in the introduction arises in the context of analytical modeling of TCP dynamics. Therefore, the reader only interested in the combinatorial aspects of the problem can focus on the remaining sections. This section explains in detail the motivation for the problem.

While a lot of attention has been given to modeling the temporal aspects of TCP traffic (see e.g. Jaiswal et al. [2]), the dynamics of packet IDs has not received the same attention. As Bennett et al. [3] have shown, packet reordering is more widespread than originally believed, and is increasingly becoming so, due to technological advances such as link striping and mobile communications. Packet reordering has many severe effects on overall traffic characteristics; hence it is an important component of TCP dynamics (we refer the reader to [3] for further discussion).

Paper [1] introduced RESTORED, a methodology for semantic compression and regeneration of large TCP traces. RESTORED is based on the following observation: TCP guarantees to deliver an ordered packet stream to the application layer and needs to buffer packets that arrive out of order. Consequently, the received packets can be classified into two types: those that could be immediately passed to the application layer, and those that have to be temporarily buffered. A received packet that allows the buffer to flush is called a *pivot packet*. All packets appearing in order are trivially pivots. RESTORED divides the received sequence into segments, bounded by pivot packets. Segments correspond to one of two *phases*:

- An *ordered phase*, in which no reordering is present; thus there is no need for buffering.
- An *unordered phase*, in which there is reordering and buffering.[1] Each occurrence of this phase ends when a pivot packet is received.

RESTORED preserves packet reordering properties of TCP traffic, up to a notion of semantic equivalence of packet traces. This notion is called *behavioral equivalence* and can be motivated as follows:

Definition 1. Let ACK_i be defined as the smallest integer that does not appear among the first i packet IDs (also, define $\mathrm{ACK}_0 = 1$). Parameter ACK_i is called *the acknowledgement* (ACK) *at stage i*.

The previous definition relies on the simplifying assumption that in the implementation of TCP each received packet is ACKed, and that value ACK_i is the only information carried by the ACK packet. Of course, real-life acknowledgment policies of TCP can be more complicated [4].

Consider now the following two packet ID sequences: 4 2 3 1 and 4 3 2 1.

These two sequences trigger identical ACK responses, namely 1 1 1 5, i.e., we arrive at the following two mappings:

$$\begin{array}{cccccccccc} 4 & 2 & 3 & 1 & \rightarrow & 1 & 1 & 1 & 5, \\ 4 & 3 & 2 & 1 & \rightarrow & 1 & 1 & 1 & 5. \end{array} \tag{1}$$

Since TCP is a *receiver-driven* protocol, assuming identical network conditions, and discounting possible differences in the value of the congestion window at the beginning of the sequences, *the two ID sequences trigger identical responses from the receiver, and should thus be regarded as indistinguishable from the standpoint of* TCP *dynamics*.

Definition 2. Two sequences of packets P and Q are *behaviorally equivalent* (written $P \equiv_{beh} Q$) if they lead to the same sequences of ACKs.

In practice one might want a notion of equivalence that is even more restrictive than behavioral equivalence. This was, for instance, the case of RESTORED. Its original motivation was to provide a way to compress TCP traces and estimate various measures of quality of service of the original traces by reconstructing "compatible" sequences. Many measures of packet reordering have been proposed in the networking literature [5–7]. Given such a measure M, one way to guarantee that sequences produced by RESTORED resemble the original sequence with respect to measure M is:

(1) Identify an equivalence notion of ID sequences $\equiv$ such that M *is consistent with respect to* $\equiv$, that is

$$(\forall A, B)\colon (A \equiv B) \Rightarrow (M(A) = M(B)). \tag{2}$$

(2) Make sure that for any sequence A, the sequence $R(A)$ regenerated by RESTORED satisfies $R(A) \equiv A$.

(See also [8] for more discussion and clarification.) Behavioral equivalence might be too coarse (as an equivalence relation) to guarantee consistency of many reordering metrics and, thus, needs to be refined. In a companion paper [9] we have considered such an equivalence notion, based on the following notion of buffer size:

[1] A technical assumption we will employ is that duplicates of packets that have already been uploaded to the application layer are discarded. This is a sensible assumption, given TCP behavior.