Packet Reordering Metrics: Some Methodological Considerations

Gabriel Istrate, Anders Hansson, and Guanhua Yan
Discrete Simulation Science (CCS-5), Los Alamos National Laboratory
P.O. Box 1663, MS M997; Los Alamos, NM 87545
Email: gabrielistrate@acm.org, {hansson, ghyan}@lanl.gov

Abstract

Characterizing what makes a packet reordering metric meaningful is a problem that has attracted significant interest, but it still lacks a universally accepted solution. We contribute to this discussion by investigating some theoretical concepts that make the following simple intuitions precise:

– A metric that is inconsistent*, i.e., gives different values on two similar TCP traces, should not be regarded as useful.*

– We formalize the notion of two traces being "identical modulo unimportant details" using similarity relations.

– If "real-life" traces differ from random sequences by always satisfying certain reorder invariants*, then we should only use traces satisfying these invariants when investigating the consistency of a reordering metric.*

We illustrate these concepts in the context of RESTORED*, an approach to semantic compression of TCP traces [10]. In particular, we discuss the consistency of two metrics defined by Jayasumana et al. [1, 12] with respect to the similarity notions defined in [8, 9, 10].*

1 Introduction

Research on network traffic modeling has recently highlighted the central role *packet reordering* plays in the overall dynamics of TCP traffic [2, 3, 13]. New measures of reordering have been introduced [17], and a variety of measures have been reviewed [14, 18]. While several criteria for the usefulness of reordering metrics have been proposed, we are still far from agreeing on a universal methodology for defining and employing reordering metrics.

Packet reordering presents a number of interesting challenges for both applied and theoretical researchers. Indeed, there exists a rich set of concepts and results in the Theoretical Computer Science literature (e.g., see [4]) that deal with reordering metrics, mostly in the context of sorting algorithms. Two of the problems this theory addresses are: (i) Given a reordering measure M, can one find a sorting algorithm that is optimal with respect to M? (ii) Given two reordering measures M_1 and M_2, is M_1 *finer than* M_2, in the sense that any M_1-optimal sorting algorithm is also M_2-optimal?

In contrast, in the network traffic context we are given an "on-line sorting" algorithm—in the form of the TCP protocol—and we are interested in gauging the relevance of the various reordering metrics for this particular algorithm. This problem is further complicated by the fact that a complete specification of the "sorting algorithm" is practically impossible, as it includes features such as network topology or background traffic that cannot be determined with complete accuracy. Nevertheless, by taking a (pre)theoretical perspective,[1] we hope to advance the characterization of what makes a reordering metric meaningful. In particular, we sketch the the following systematic approach:

(i) We advocate the use of *similarity relations* as a formal way to specify all the information we deem important in a given measurement context.

(ii) We define the notion of *consistency* of a reordering metric with respect to a similarity relation, and argue that, provided all the important information has been encoded in the similarity relation, useful metrics should be consistent. We further discuss the consistency of some of the metrics from [14] with respect to various equivalence notions.

(iii) We highlight the concept of a *reorder invariant*, and illustrate this notion by a metric called SUS from the literature on sorting algorithms [4].

The concepts in this paper were motivated by an approach, called RESTORED [10], for receiver-oriented modeling and compression of large TCP traces, in a

[1] That is, we are primarily concerned with modeling and (when possible) defining precise mathematical concepts, rather than proving theorems.

0-7695-2622-5/06/$20.00 (c) 2006 IEEE

manner that preserves some of the semantics of the TCP protocol and incorporates information on packet dynamics. To make the paper self-contained, a brief introduction to RESTORED is given next.

2 An Introduction to RESTORED

RESTORED is a receiver-oriented model of large TCP traffic that incorporates information on the dynamics of packet IDs. It can be used to estimate QoS measures offline. Rather than running a large number of such measurements in an on-line fashion, we first "compress" the trace into a small "sketch." If needed we can perform a large number of measurements on the regenerated trace.

In a nutshell, RESTORED is a multi-scale model that uses a Markov chain to model the dynamics of traffic at large scales. This is consistent with results that show that all the correlations in the temporal structure of network traffic disappear above a certain time scale [7]. To describe this approach, we make the simplifying assumption that all packets have identical payload. This allows a bijective mapping from TCP sequence numbers to packet ID numbers, with the convention that the smallest sequence number is mapped to ID 1. In the present exposition, a *connection* consists of a sequence of packet IDs, together with positive real numbers corresponding to *packet arrival times* (this is the only information in a real trace that we attempt to capture). Suppose the receiver observes the following packet stream (where we only display the ID numbers of the packets, and not their arrival times)[2]

$$\underbrace{1\ \ 2\ \ 3}_{\mathcal{O}}\ \ \underbrace{5\ \ 6\ \ 7\ \ 4}_{\mathcal{U}}\ \ \underbrace{8\ \ 9\ \ 10}_{\mathcal{O}}\ \ \underbrace{12\ \ 13\ \ 14\ \ 11}_{\mathcal{U}} \quad (1)$$

The order of the IDs corresponds to the order in which packets arrive at the destination, and in our example, we see that packets 4 and 11 arrive out of order. Since TCP guarantees to deliver an ordered packet stream to the application layer, it follows that there is a need for buffering of certain packets. One can, consequently, classify the received packets into two types: those that can be immediately passed to the application layer, and those that are temporarily buffered before delivery. In our example, packets 5, 6, and 7 are temporarily buffered, and the buffer cannot be flushed until packet 4 is received. Likewise, packets 12, 13, and 14 are temporarily buffered, and the buffer is flushed at the arrival of packet 11. A packet that marks the end of a sequence of consecutively buffered packets will be called a *pivot packet*. Also, packets that are immediately delivered to the application layer are trivially pivots. In our example, packets 1, 2, 3, 4, 8, 9, 10 and 11 are thus all pivots.

[2]We omit the slow start phase from our discussions.

This definition suggests a coarsened view of TCP with two states: An *ordered state* $\mathcal{O}$, in which packets arrive in order, and an *unordered state* $\mathcal{U}$ in which there is reordering and buffering. Each occurrence of State $\mathcal{O}$ is followed by one or more occurrences of State $\mathcal{U}$. Now, explicitly incorporating time as well, one can thus define a sequence of triples (s_1, p_1, t_1), (s_2, p_2, t_2), $\ldots$, where $s_n \in \{\mathcal{O}, \mathcal{U}\}$ is the state descriptor, $p_n > 0$ is the number of packets received in state s_n, and $t_n > 0$ is the time spent in state s_n. In [10] we have provided statistical evidence that a Markov chain is often sufficient for capturing the dynamics of this sequence of triplets. The sequence of packet IDs in the ordered state is trivial by definition. In the unordered state we obtained further compression by a many-to-one mapping $\mathcal{M}$ of sequences of IDs into "sketches." The mapping $\mathcal{M}$ can be inverted in polynomial time. This is used in the regeneration algorithm, where we regenerate a trace by simulating the macroscopic Markov chain; when in the unordered state we first sample a sketch S from the definition of such sketches and then reconstruct a sequence of IDs that maps to S. The many-to-one mapping needs to be defined with respect to the particular reordering metrics we want preserved. In [8] and [9] we provide two such variants (discussed below). The outlined mechanism can be completed to generate arrival times in order to reconstruct a full trace.

3 Modeling Context with Similarity Relations

One problem raised by TCP models such as RESTORED is that we want a formal way to guarantee that the reconstructed sequences are "similar" to the original one. Of course, one could simply try to solve this problem empirically by comparing original and reconstructed sequences with respect to a few reordering metrics (paper [10] reports such results for three such metrics, RD and RBD from [17] as well as a metric called inversion spectrum, defined as the distribution of inversion displacements). However, the choice of metrics for such a comparison is somewhat ad-hoc, and there seems to be no principled way to choose "good benchmark measures."

In general, as discussed in [17], *the quality of a measure of reordering can only be evaluated* **in the context of an application***: for instance, if the application is able to tolerate a certain amount of reordering then the exact amount of reordering is not essential, as long as it is within those bounds.*

We propose a (somewhat more) principled approach. In a nutshell, it can be briefly described as follows: **one should attempt to formally specify the "context of a certain application" (and the trace prop-**

erties we deem important) by a *similarity relation* R on traces. For instance, notions such as "the throughput (number of inversions, etc.) of the two traces differs by no more than 5%" provides natural examples of similarity relations.

But what is a similarity relation (of objects) in general? It is hard to give an all-encompassing definition, and we will not attempt to do so, though definitions of this type arise in several areas of science, such as economics [19], rough set theory [15], fuzzy set theory (e.g., [21]), clustering in linguistics and biology [20], etc. A similarity relation needs to be *reflexive* [i.e. for all A, $R(A, A)$], though it is not necessarily symmetric. Nevertheless, some natural similarity relations are *equivalence relations*, that is *reflexive*, *symmetric* [$R(A, B) \Rightarrow R(B, A)$ for all A, B] and *transitive* [$R(A, B) \wedge R(B, C) \Rightarrow R(A, C)$ for all A, B, C].

To motivate the notion of trace similarity that is most relevant for results in [10], consider the following two hypothetical sequences of packet IDs: A = (1 2 3 4 5 6 10 9 8 7), and B = (1 2 3 4 5 6 10 8 9 7). Assume that the TCP implementation uses simple ACKs (as opposed to SACK), and acknowledges every single packet. Then *the two sequences will generate on the receiver side* **the same** *sequence of ACKs*, namely 2 3 4 5 6 7 7 7 7 11. Since in TCP applications it is the receiver ACK sequence that drives the dynamics of the congestion window, **assuming identical network conditions for the two ACK sequences, the two traces can be regarded as "equivalent,"** from a receiver-oriented standpoint. We thus arrive at the following definition:

Definition 1 *Two sequences of packets A, B are* behaviorally equivalent *if they yield the same sequences of ACKs.*

It is easy to see that behavioral equivalence is indeed an equivalence relation on TCP traces. Of course, in a different scenario the requirements of the application at the top of the protocol stack (e.g., how much loss/packet reordering the application is willing to tolerate), the particular details of the TCP implementation, etc., might impose a different notion of "semantic equivalence" of TCP traces. For instance, none of the aspects related to the *arrival times* of packets is captured by behavioral equivalence. If we employ mechanisms for keeping track on the receiver side of the RTT (as provided by [11]), then packet arrival times can convey semantic information about the TCP flow. This example, however, provides a blueprint for a systematic approach:

(1) **Precisely specify a similarity relation $\equiv$ on the set of possible TCP traces. The relation $\equiv$ should capture all the aspects we deem important**. It can incorporate features of the semantics of the TCP protocol, as well as features of the specific application (video, etc.) at the top of the protocol stack.

(2) **A model M of TCP traffic should come together with a similarity notion R and should provide a guarantee that $R(A, M(A))$ for any real TCP trace** A (where $M(A)$ is the trace regenerated by M from input A).

4 Consistency of Reordering Metrics

Not all reordering metrics are expected to be captured by an approach such as RESTORED. In what follows we introduce a useful restriction on the reordering metrics. To motivate this restriction, consider the following metric, called n-reordering, proposed in [14] and defined as follows:

Definition 2 *Let $s[1], s[2], \ldots s[l]$ be a stream of packet IDs as received at a destination. For $n \geq 1$ a packet $s[i]$ is* n-reordered *if for all $i - n \leq j < i$ we have $s[j] > s[i]$. The* degree of n-reordering *of a connection is*

$$S(n) = \frac{number\ of\ n\text{-}reordered\ packets}{number\ of\ received\ packets}$$

Also define $R(n) = S(n) - S(n+1)$. The reordering distribution *is the probability distribution $R(n)$, $n \geq 1$.*

Despite being behaviorally equivalent, the two sequences A and B defined in the previous section have different reordering distributions, $R_A(0) = 0.7, R_A(1) = 0.1, R_A(2) = 0.1, R_A(3) = 0.1, R_A(i) = 0, i \geq 4$, and $R_B(0) = 0.8, R_B(1) = 0.1, R_B(3) = 0.1$, $R_B(i) = 0$ for all other i, respectively. Thus a reconstruction method that only guarantees that the reconstructed sequence is behaviorally equivalent to the original one cannot be expected to recover the reordering distribution. This motivates the following definition:

Definition 3 *A reordering metric M is* consistent with respect to a similarity relation R *if for all sequences A, B $R(A, B) \Rightarrow (M(A) = M(B))$.*

Conclusion 1 If *all* the relevant information about a connection is encapsulated in a similarity relation R then, obviously, a metric that is inconsistent with respect to R is *not* an adequate reordering metric.

Definition 4 *Similarity relation R_2* refines relation *R_1 ($R_2 \subseteq R_1$) if $\forall A, B$ $[R_2(A, B) \Rightarrow R_1(A, B)]$.*

The following is now a simple observation:

Theorem 1 *Let M be a reordering metric and R_1 and R_2 similarity relations such that $R_2 \subseteq R_1$. If M is consistent with respect to R_1 then M is consistent with respect to R_2.*

Proof. Let A and B be traces and suppose $R_2(A, B)$. Then $R_1(A, B)$ since M is consistent w.r.t. R_1, $M(A) = M(B)$. But this means that M is consistent w.r.t. R_2. □

Theorem 1 highlights the trade-off between the tightness of similarity measures and the consistency of reordering metrics: **the tighter the metric, the more measures are consistent**. A general lesson is

Conclusion 2 When defining the similarity relation we should attempt to define it in the most restrictive way possible that is compatible with the requirements of the scenario at hand.

The number of metrics consistent under $\equiv_{beh}$ is rather limited. We have therefore studied two other similarity relations. The first one [9] is defined as:

Definition 5 *Let $A = \{A_1, A_2, \ldots, A_n\}$ be a sequence of packet IDs. We define an operator* B *that after receiving a packet A_i at time index i outputs the size of the buffer needed to store all out-of-order packets up to stage i (we assume that packets that are in-order are immediately evicted from the buffer). Two sequences of packets P and Q are* buffer equivalent *($P \equiv_{buf} Q$) if $Buf(P) = Buf(Q)$.*

Our last equivalence notion is [8]:

Definition 6 *Let $A = \{A_1, A_2, \ldots, A_n\}$ be a sequence of packet IDs. We define the* FB *as an operator that after receiving a packet A_i at time index i, outputs the difference between the highest ID (H_i) seen so far and the highest ID (L_i) that could be uploaded,* $\mathrm{FB}(A_i) = H_i - L_i$. *In other words,* FB *is the size of the smallest buffer big enough to store all packets that arrive out of order, where the definition of size accounts for reserving space for unreceived packets with intermediate IDs as well.* Full Buffer sequence $FB(P)$ *associated with a sequence P of packet IDs is the time-series of FB values computed after each packet has been received. Sequences P and Q are* FB equivalent *($P \equiv_{FB} Q$) if $FB(P) = FB(Q)$.*

Equivalences we have defined so far are sensitive to losing packets. Their definitions can be, however, modified to take into account losing packets as well. This is not important for our application, since in RESTORED we only consider completed occurrences of the unordered state. Maps FB and Buf can be inverted in polynomial time ([8, 9]). Therefore, when employed in RESTORED, the reconstructed sequences of packet IDs corresponding to one unordered state are equivalent (w.r.t. the respective equivalence notions) to some sequence of IDs corresponding to an unordered state of the original sequence. Therefore, all reordering metrics that are consistent with respect to these equivalences are going to be closely estimated with high probability by the reconstruction process. Equivalence $\equiv_{FB}$ is indeed a refinement of $\equiv_{beh}$:

Proposition 1 *([8]) Suppose that the receiver uses simple ACKs and acknowledges every packet. Then any FB equivalent sequences A and B are also behaviorally equivalent.*

On the other hand buffer equivalence, though seemingly more natural, is *incomparable* with $\equiv_{beh}$:

Theorem 2 *There exist sequences A, B, C and D with $A \equiv_{buf} B$ but $A \not\equiv_{beh} B$, $C \equiv_{beh} D$ but $C \not\equiv_{buf} D$.*

Proof. An example is $A =$ 2 3 3 1 and $B =$ 3 4 1 2. They both map via Buf to sequence 1 2 2 0, hence they are buffer equivalent, but they are not behaviorally equivalent, since they yield ACK sequences 1 1 1 4 and 1 1 2 5, respectively. Also, let $C =$ 2 2 3 4 1 and $D =$ 2 3 2 4 1. They both yield ACK sequence 1 1 1 1 5, but different buffer sequences 1 1 2 3 0 and 1 2 2 3 0, respectively. □

Buffer equivalence is only guaranteed (see [9]) to be a refinement of behavioral equivalence for *permutations* (i.e. sequences with no repeats and no lost packets). Finally, equivalences $\equiv_{buf}$ and $\equiv_{FB}$ are incomparable:

Theorem 3 *There exist sequences of IDs A, B, C and D with $A \equiv_{buf} B$ but $A \not\equiv_{FB} B$, $C \equiv_{FB} D$ but $C \not\equiv_{buf} D$.*

Proof. Sequences $A =$ 2 4 3 1 and $B =$ 2 3 4 1 yield the same buffer sequence 1 2 3 0, but different FB sequences 2 4 4 0 and 2 3 4 0, respectively. Sequences $C =$ 4 2 2 3 1 and $D =$ 4 2 3 2 1 yield the same FB sequence 4 4 4 4 0, but different buffer sequences 1 2 2 3 0 and 1 2 3 3 0, respectively. □

5 Consistency of Reordering Metrics: Some Examples

The first objective of this section is to provide natural illustrations for the concept of consistency, by pro-

viding examples of reordering metrics consistent under one of the two equivalence metrics $\equiv_{FB}$ and $\equiv_{buf}$. These examples show that the incomparability of these equivalence metrics also translates into the incomparability of the sets of consistent reordering metrics.

Observe that when all packets have the same payload p, FB is linearly related to the size of the advertised window by the following relation ([16], p. 385)

$$\text{AdvertisedWindow} = \text{MaxRcvBuffer} - p \cdot \text{FB}. \quad (2)$$

Therefore the following result is true:

Theorem 4 *Define $AdvertisedWindow(W)$ to be the operator that maps packet sequence W into the time series consisting of the values of the parameter AdvertisedWindow when receiving packet sequence W. Then any reordering metric M that can be expressed as $M(W) = f(AdvertisedWindow(W))$ for some function f (e.g., the average advertised window) is consistent under $\equiv_{FB}$. This latter measure is not consistent under $\equiv_{buf}$.*

Proof. The first statement follows directly from equation (2) since if $FB(P) = FB(Q)$ then AdvertisedWindow(P) = AdvertisedWindow(Q). For the second statement take $R = (2\ 3\ 4\ 5\ 1)$ and $S = (5\ 4\ 3\ 2\ 1)$. Then $Buf(R) = Buf(S) = (1\ 2\ 3\ 4\ 0)$ but $FB(R) = (2\ 3\ 4\ 5\ 0)$, and $FB(S) = (5\ 5\ 5\ 5\ 0)$. The last two equalities immediately imply the fact that average values of parameter $AdvertisedWindow(R)$ and $AdvertisedWindow(S)$ are equal to MaxRcvBuffer $- 14p/5$ and MaxRcvBuffer $- 4p$, respectively, which are different for $p \neq 0$. □

On the other hand, we have

Theorem 5 *Any reordering metric M that can be expressed as $M(W) = f(Buf(W))$ for some function f (e.g., the average value of the buffer size) is consistent under $\equiv_{buf}$ equivalence, This latter measure is inconsistent under $\equiv_{FB}$ equivalence.*

Proof. The first statement follows by the definition of $\equiv_{buf}$. For the second statement take the two sequences $C = 4\ 2\ 2\ 3\ 1$ and $D = 4\ 2\ 3\ 2\ 1$. Then, as we saw in Theorem 3 $C \equiv_{FB} D$, but $M(C) = 8/5$, $M(D) = 9/5$. □

Finally, we discuss the consistency of two metrics from [1, 12] (see also discussions in [18, 17]). For brevity, we do not present definitions of these measures, but refer the reader to [12].

Reorder Density (RD) is *not* consistent under buffer, behavioral, or FB equivalence: sequences of IDs 4 2 3 1 and 4 3 2 1 have different RD distributions but they are buffer, behaviorally, and FB equivalent. Reorder Buffer-Occupancy Density (RBD) is *not* consistent under buffer equivalence. This is witnessed by sequences 2 3 3 1 and 3 4 1 2. On the other hand one can prove that RBD is consistent with respect to behavioral (and FB) equivalence. The proof of this result is slightly more involved, and will be presented in the journal version of this paper. This version will also contain a classification of the consistency of metrics from [14] with respect to the three equivalence notions considered in this paper.

Conclusion 3 The precise definition of the similarity notion greatly impacts consistency. Different similarity notions can have incomparable sets of consistent metrics.

6 Reorder Invariants

The consistency criterion in the previous section can be discussed as being too pessimistic: a measure M is inconsistent with respect to a similarity notion $\equiv$ if there exist two (hypothetical) sequences of packet IDs that are equivalent with respect to $\equiv$ but differ in the value of M. This criterion does not take into account the fact that not all possible ID sequences appear in real-life traces. Therefore some measures that are inconsistent in theory could be consistent "in practice." This is actually the case of metric RD, as shown in [10]. One way to address this problem is to build a set of reordering patterns arising from real traces and investigate the consistency of the various measures on this set of patterns. We will illustrate this approach in an extended version of the paper. This solution, however, has the drawback that the selection of traces is somewhat arbitrary. In the remainder of this section we suggest a different approach: the use of *reordering invariants* to reduce the set of traces considered in the definition of consistency. In a nutshell the idea is the following: TCP attempts to preserve packet sequence integrity. Therefore measures of reordering of real traffic data fall into two types (i) those that have values roughly similar to those of a sorted sequence, reflecting the order-preserving nature of the TCP protocol, and (ii) those that can vary significantly. We call measures of the first type *reorder invariants*. Reorder invariants are interesting since they expose limits on the nature of TCP traces (which could potentially be used for the definition and estimation of entropy-like notions associated to sequences of packet IDs). A reorder invariant

0-7695-2622-5/06/$20.00 (c) 2006 IEEE

D could be used as follows: if we assume that real-life ID sequences X satisfy $D(X) \leq k$ then, to test for the inconsistency of measure M: (i) Generate a set of random sequences of IDs $Y_1, \ldots Y_r$ with $D(Y_i) \leq k$. (ii) Test whether there exist two sequences Y_i and Y_j such that $Y_i \equiv Y_j$ and $M(Y_i) \neq M(Y_j)$.

Clearly a number of technical issues (finding examples of reorder invariants D, determining numbers k and r, being able to randomly sample strings X with $D(X) \leq k$) need to be solved before the procedure we outlined could work. We do not attempt to do this now. Instead, we only present a metric that could be a reorder invariant. The metric is called **shuffled up-sequences (SUS)** [**4**]. It is defined as *the minimum number of ascending subsequences into which we can partition each listed sequence of packets.*

We have computed the value of the SUS metric on a set of real packet traces collected during August 2001 at the border router of the Computer Science Department, University of California, Los Angeles (UCLA), CA. The set was obtained by the UCLA Network Research Lab and modified for public use by the UCLA Laboratory for Advanced Systems Research. In particular, we have used the five TCP traces, that were available on-line at the start of our work. They are labeled TRACE5, TRACE7, TRACE8, TRACE9, and TRACE10. We parsed the traces into different connections and found that most of the connections are very short. As an example, TRACE7 consists of 245,718 connections, but 60% of them contain only one or two packets, 80% contain 10 packets or less, and 98% contain at most 100 packets. This is, of course, in line with the common observation that a small percentage of the data flows accounts for a large percentage of the total traffic [5, 6]. Since our aim is to highlight nontrivial network dynamics, we have chosen to study only connections with at least 100 packets. Still, the traces are large enough to allow meaningful analysis: each traces contains at least 3399 such "long" connection. We partitioned the connections into sequences of IDs corresponding to the ordered/unordered state and computed the SUS metric for *all* sequences corresponding to the unordered state. Over 95% of them are of type SUS = 2. Complete results are presented in Table 1.

To test that the SUS measure remains relatively constant even when we induce significant reordering in the network, we used the ns-2 network simulator to simulate a dumbbell topology with two routers and six end-hosts (see Figure 1).

Each link that connects an end-host and a router is duplex and has bandwidth 10 Mbps and propagation delay 1 millisecond. We use two simplex links in reverse directions to connect the two routers. Both of them have propagation delay 1 millisecond, but the one from router r1 to r2 has bandwidth 2 Mbps and the other has bandwidth 10 Mbps. The relatively low bandwidth from r1 to r2 is intended to create congestion at r1. In the network, we have three TCP flows that transfer data from end-hosts s1 to t1, s2 to t2, and s3 to t3, respectively. We perform two sets of experiments. In the first one, every link uses a drop-tail queue, which is a FIFO buffer dropping packets at its tail when it overflows. In the second one, all the links use drop-tail queues except the one from router r1 to r2. This special link uses a queue that still drops packets at its tail when it overflows, but on each new packet arrival, it randomizes the order of all the packets in the buffer.

We have computed the sequences of IDs corresponding to the occurrences of the unordered state in all three connections, and computed the distribution of SUS values of these patterns. The results are presented in Table 2. Even though the experiment introduces a significant amount of packet reordering, it does so by changing the nature of reordering patterns, rather than their overall number. In particular, the second experi-

Table 1. Distribution of SUS in real data

SUS	TR.5 %	TR.7 %	TR.8 %	TR.9 %	TR.10 %
2	95.072	96.876	98.223	95.293	98.967
3	4.542	2.291	1.669	4.324	0.953
4	0.328	0.166	0.094	0.306	0.070
5	0.035	0.026	0.010	0.057	0.006
6	0.017	0.005	0.004	0.011	0.003
7	0.002	0.002	—	0.002	—
8	0.002	0.004	—	0.0009	—
9	0.003	—	0.0007	—	—
10	—	—	—	0.002	0.001
11	—	—	—	0.002	—
12	—	—	0.0007	—	—
13	—	—	—	0.0009	—
14	—	—	—	—	—
15	—	—	—	—	—
16	—	—	—	0.0009	—

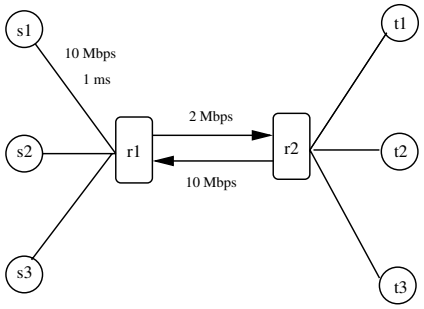

Figure 1. Simulated network topology.

Table 2. Distribution of SUS in ns-2 data

Run	SUS = 2	SUS = 3	SUS = 4
1st	38,757 (99.9%)	1 (0%)	2 (0%)
2nd	25,686 (67.7%)	10,416 (27.4%)	1762 (4.6%)
Run	SUS=5	SUS= 6	Patterns
1st	—	—	38,764
2nd	83 (0.2%)	2 (0%)	37,949

ment creates a larger number of patterns. Still even in this case the vast majority of reordering patterns have SUS = 2. Furthermore, the largest observed value of the SUS parameter only increases from 4 to 6. This means that the TCP algorithm does manage to keep the SUS parameter small even when facing significant reordering.

7 Conclusions

Our work has highligted some concepts that we believe are useful in studying packet reordering metrics: *similarity relations*, *consistency* of a metric, and *reordering invariants*. We have illustrated these concepts with natural notions of similarity and investigated the consistency of two metrics defined by Jayasumana *et al.* [12]. This work has been supported by the U.S. Department of Energy under contract W-705-ENG-36.

References

[1] T. Banka et al. Metrics for degree of reordering in packet sequences. In *Proc. 27th IEEE LCN*, pages 333–342, 2002.

[2] J. Bellardo and S. Savage. Measuring packet reordering. In *Proc. ACM SIGCOMM Internet Measurement Workshop*, pp. 97–105, 2002.

[3] J. C. R. Bennett et al. Packet reordering is not pathological network behavior. *IEEE/ACM Transactions on Networking*, 7(6):789–798, 1999.

[4] V. Estivill-Castro and D. Wood. A survey of adaptive sorting algorithms. *ACM Computing Surveys*, 24(4):441–476, 1992.

[5] W. Fang and L. Peterson. Internet-AS traffic patterns and their implications. In *Proc. IEEE GLOBECOM Conf.*, pp. 1859–1868, 1999.

[6] A. Feldmann et al. Deriving traffic demands for operational IP networks: Methodology and experience. *Proc. ACM SIGCOMM*, pp. 257–270, 2000.

[7] D. R. Figueiredo et al. On TCP and self-similar traffic. *Perf. Evaluation*, 61(2-3):129–141, 2005.

[8] A. Hansson, G. Istrate, and S. Kasiviswanathan. Combinatorics of TCP reordering. *Journal of Combinatorial Optimization*, special issue on Network Applications (to appear), 2006.

[9] G. Istrate and A. Hansson. Counting preimages of TCP reordering patterns. submitted to Discrete Applied Mathematics, October 2005.

[10] G. Istrate, A. Hansson, S. Thulasidasan, M. Marathe, and C. Barrett. Semantic compression of TCP traces. In *Proceedings of the IFIP NETWORKING'06*, vol. 3976 of *Lecture Notes in Computer Science*, pp. 123–135. Springer, 2006.

[11] S. Jaiswal et al. Inferring TCP connection characteristics through passive measurements. In *Proc. IEEE INFOCOM*, 2004.

[12] A. P. Jayasumana et al. Reorder density and reorder buffer-occupancy density - metrics for packet reordering measurements. IETF draft, http://cnrl.colostate.edu/Reorder/draft-jayasumana-reorder-density-06.txt.

[13] M. Laor and L. Gendel. The effect of packet reordering in a backbone link on application throughput. *IEEE Network*, 16(5):28–36, 2002.

[14] A. Morton et al. Packet reordering metric for ippm. IETF draft, available from http://www.ietf.org/internet-drafts/draft-ietf-ippm-reordering-09.txt. Accessed Sept. 2005.

[15] Z. Pawlak. *Rough Sets*. Kluwer, 1991.

[16] L. Peterson and B. S. Davie. *Computer Networks. A Systems Approach*. Morgan Kauffman, San Francisco, CA, 2nd edition, 2000.

[17] N. M. Piratla et al. RD: A formal, comprehensive metric for packet reordering. In *Proc. IFIP Networking 2005*, vol. 3462 of *Lecture Notes in Computer Science*, pp. 78–89. Springer Verlag, 2005.

[18] N. M. Piratla et al. A comparative analysis of packet reordering metrics. In *Proc. COMSWARE'06*.

[19] A. Rubinstein. Similarity and decision-making under risk. *J. Economic Theory*, 46:145–153, 1988.

[20] D. Sankoff and J. Kruskal. *Time Warps, String Edits, and Macromolecules: The Theory and Practice of Sequence Comparison*. Addison Wesley, 1983.

[21] L. Zadeh. Similarity relations and fuzzy set ordering. *Information Sciences*, 3:177–200, 1970.