manuscript No.
(will be inserted by the editor)

Packet Reordering Metrics: Some Methodological Considerations

Gabriel Istrate · Anders Hansson · Guanhua Yan

January 8, 2010

Abstract Characterizing what makes a packet reordering metric useful in practice is a interesting engineering problem in computer networking that is largely application-dependent, which has so far only received somewhat ad-hoc solutions.

We propose a methodology for adressing this issue based on

1. the systematic specification of "important details" of application contexts via *similarity relations*.
2. The use of *the consistency of metrics* with respect to similarity relations on traces as a threshold that useful metrics need to pass.
3. The investigation and systematic use of *reorder invariants*, to circumscribe "real-life" properties of network traffic and make the theoretical notion of consistency more adequate in practice.

We illustrate our methodology using a combination of rigorous results and experiments, as follows:

1. We prove several mathematical results dealing with the properties of several similarity notions we consider.
2. We also rigorously address the issue of consistency of several measures of reordering used in practice. In particular, we discuss the consistency of two metrics defined by Jayasumana *et al.* [BBJ02, JPB+08] with respect to the similarity notions defined in [HIK06, IH08, IHT+06].
3. We present experimental evidence that a particular reordering metric called *Shuffled Up Sequences (SUS)* is a reordering invariant.

This paper is the corrected and substantially revised final version of an extended abstract that has appeared in the Proceedings of the Second International Conference on Networking and Services (ICNS'06).

Gabriel Istrate
eAustria Research Institute, V. Pârvan 4, cam 045B, Timişoara, RO 300223, Romania E-mail: gabrielistrate@acm.org

Anders Hansson
Ericsson, Gőteborg, Sweden E-mail: ahansson@gmail.com

Guanhua Yan
CCS-3, Los Alamos National Laboratory, Los Alamos, NM 87545, E-mail: ghyan@lanl.gov

4. Finally, we show that one of the metrics in [BBJ02], though theoretically inconsistent, is consistent given the previously observed reorder invariant.

We discuss and illustrate these concepts in the context of RESTORED, an approach to semantic compression of TCP traces. [IHT+06].

Keywords: TCP, reordering measures, combinatorial problems.

1 Introduction

Research on traffic modeling in computer networking has recently highlighted the central role *packet reordering* plays in the overall TCP traffic dynamics [BS02, BPS99, LG02]. New measures of reordering have been recently introduced [PJB05], and a variety of measures have been formally reviewed [MCR+06]. While several criteria for the usefulness of reordering metrics have been previously discussed in the literature [PJ08], the field practice still relies on imprecise, somewhat ad-hoc solutions. In this sense, approaches to solving this problem have a heuristic nature (although for a somewhat unorthodox application of the term heuristic - the difficulty relies in precisely specifying good quality criteria, rather than in finding methods for optimally solving a problem).

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 [ECW92]) 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 context of network traffic we are given the "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**. The 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.

In this paper we take a modeling perspective (that is, we are primarily concerned with isolating precise concepts useful to this problem) hoping to advance the state of the art in deciding 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 [MCR+06] 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 sorting algorithms literature [ECW92]. Reorder

invariants can be used to make the traces employed in the definition of consistency resemble "real" traces.

The concepts and methodology outlined in this paper were motivated by evaluating the performance of RESTORED [IHT+06], a method for receiver-oriented modeling and compression of large TCP traces, that incorporates information on the dynamics of packet IDs. RESTORED 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. A natural question, that motivates the concepts in this paper, is what measures of reordering can an approach such as RESTORED be able to recover.

A final word about the significance of our result. In the following section we give several precise definitions and prove results about the concepts we introduced. While (hopefully) realistic enough, from a traffic modeling standpoint, these results are mainly pedagogical, and serve to illustrate the methodology we put forward. As such, we do not view our main contribution as being primarily mathematical, but conceptual.

2 Context of our work: RESTORED

In a nutshell, RESTORED is a multi-scale model that uses a Markov chain to model the dynamics of traffic at large scales. To describe it 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)[1] $\underbrace{1\ \ 2\ \ 3}_{\mathcal{O}}\ \underbrace{5\ \ 6\ \ 7\ \ 4}_{\mathcal{U}}\ \underbrace{8\ \ 9\ \ 10}_{\mathcal{O}}$ $\underbrace{12\ \ 13\ \ 14\ \ 11}_{\mathcal{U}}$. 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. 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 $\mathcal{U}$. The sequence

[1] We omit slow start from discussion.

of packet IDs in the ordered state is trivial. In the unordered state we employ a many-to-one mapping $\mathcal{M}$ of sequences of IDs into "sketches." to achieve extra compression. The mapping $\mathcal{M}$ can be inverted in polynomial time and is used in the regeneration algorithm: 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 [HIK06] and [IH08] we provide two variants. The outlined mechanism can be completed to generate arrival times in order to reconstruct a full trace.

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. One could simply solve this problem experimentally by comparing original and reconstructed sequences with respect to a few reordering metrics. Paper [IHT$^+$06] takes this route for three such metrics, RD and RBD from [PJB05] as well as a third metric called inversion spectrum. However, the choice of metrics for such a comparison is quite ad-hoc, and there seems to be no principled way to choose "good benchmark measures."

We propose a (somewhat more) principled approach. Our starting point is the observation [PJB05], that *the quality of a performance measure 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.* The first step of our approach is thus **attempt to formally specify the "context of a certain application" (and the trace properties 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%" provide natural examples of similarity relations.

But what is a similarity relation (of objects) in general? It is hard to give a general definition, and we do not attempt to do so, though definitions of this type arise in several areas of science, such as economics [Rub88], fuzzy logic [Zad70], clustering in linguistics and biology [SK83], etc. A similarity relation should be *reflexive* though it is not necessarily symmetric. Nevertheless, many natural similarity relations are *equivalence relations.*

To motivate the notion of trace similarity relevant for the experimental results in [IHT$^+$06], 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 packet sequences A, B are *behaviorally equivalent* if they yield the same sequence 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 might require a different notion of "semantic equivalence". For instance, none of the aspects related to the *arrival times* of packets is captured by

behavioral equivalence. 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).

3 Consistent Metrics

Our experience with RESTORED shows that we cannot expect to recover all measures of reordering. Sometimes the reasons are structural. Consider for example the following metric, called n-reordering, proposed in [MCR+06] 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 $S(n) = \frac{\text{number of n-reordered packets}}{\text{number of received packets}}$. 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. 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 a metric 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 $\forall A, B$ $[R_2(A, B) \Rightarrow R_1(A, B)]$.

An easy result is:

Lemma 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.*

Lemma 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**.

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 thus studied two other similarity relations. The first one [IH08] is defined as:

Definition 5 Let $A = \{A_1, \ldots, A_n\}$ be a sequence of packet IDs. We define an operator Buf 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 [HIK06]:

Definition 6 Let $A = \{A_1, A_2, \ldots, A_n\}$ be a sequence of packet IDs. Define 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 ([HIK06, IH08]). 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 *([HIK06]) 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 1 *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 to be a refinement of behavioral equivalence for *permutations* (i.e. sequences with no repeats and no lost packets). This was proved in [IH08]. In fact a stronger result holds:

Theorem 2 *Let A, B be permutations of length n. Then $A \equiv_{beh} B$ iff $A \equiv_{buf} B$.*

Proof One direction, as mentioned, has been proved in [IH08]. So, to complete the proof, consider A, B permutations such that $A \equiv_{beh} B$. We have to prove that $A \equiv_{buf} B$.

We will try to reconstruct the sequence of buffer sizes Buf_i from the (common) ACK sequence for A and B. Enduring uniqueness of the construction will prove behavior equivalence.

Consider a stage index i. There are two cases:

- **Case 1:** $ACK_i = ACK_{i-1}$. Then the i'th packet is out of order. Since we are dealing with permutations, buffer size increases by one, that is $Buf_i = Buf_{i-1} + 1$.
- **Case 2:** $ACK_i > ACK_{i-1}$. Then it must be that all packets with index $ACK_{i-1} + 1, ACK_{i-1} + 2, \ldots, ACK_i - 1$ have been received prior to stage i and were buffered thus far, and packet i has index ACK_{i-1}. So the buffer shrinks in size by $ACK_i - ACK_{i-1}$.

□

Finally, equivalences $\equiv_{buf}$ and $\equiv_{FB}$ are incomparable:

Theorem 3 *There exist ID sequences 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. □

4 Consistency of Reordering Metrics: Some Examples

The first objective of this section is to provide natural illustrations for the concept of consistency, by providing examples of reordering metrics consistent under one of the equivalence metrics $\equiv_{FB}$ and $\equiv_{buf}$.

For our first examples we need several parameters, denoted by *LastByteRcvd*, *LastByteRead*, *RcvWindow*, *RcvBuffer*, respectively, that describe the dynamics TCP protocol. Their abbreviated names are quite suggestive of their meanings; one can find precise definitions in [KR03].

We can relate FB to these parameters by noting that if all packets have the same payload p, $LastByteRcvd - LastByteRead$ is p times the value of FB. Therefore a relation in [KR03] (pp. 246-247) implies the fact that FB is related to the RcvWindow parameter as follows:

$$\text{RcvWindow} = \text{RcvBuffer} - p \cdot \text{FB}. \tag{1}$$

This motivates the following result:

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

Proof. The first statement follows directly from equation (1) since if $FB(P) = FB(Q)$ then RcvW(P) = RcvW(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 $RcvW(R)$ and $RcvW(S)$ are equal to MaxRcvBuffer$-14p/5$ and MaxRcvBuffer$-4p$, respectively, 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 [BBJ02, PJBB07] (see also [PJB05, JPB$^+$08]).

Definition 7 Let π be a sequence of packet IDs. Metrics RD, RBD are defined using a threshold value $W > 0$ (called *displacement threshold* in [BBJ02], and denoted there by D_T). If a packet does not arrive within W positions from its expected location, it is assumed to be lost and removed from further consideration. These are in addition of ID's in the range $\min_i\{\pi[i]\}$ to $\max_i\{\pi[i]\}$ that are not a value of mapping π. We next remove duplicate packets. Consequently we are left with a vector $\pi[i_1], \ldots, \pi[i_m]$ of packet IDs corresponding to packets not lost or duplicated. Let $\chi[1], \ldots, \chi[m]$ be the list of such packet IDs, in sorted order.

The *displacement* $D[\chi_j]$ of the packet of ID $\chi[j]$ is the difference $\pi[j] - \chi[j]$. In other words, the displacement measure the earliness/lateness of a packet with respect to its expected arrival time in sorted order. We will present the vector of displacements *in the increasing order of packet ID's.*

Example 1 Consider packet sequence 2 3 4 1 with $W = 3$. No packet is lost under this circumstance and vector π is simply the same sequence. Vector χ is 1 2 3 4, and the displacement vector is $3\ -1\ -1\ -1$.

Definition 8 Metric RD_W is defined [BBJ02] as the distribution of relative frequencies of displacements of sequence π. RBD on the other hand [PJBB07] is defined by applying operator Buf to sequence $\pi[i_1], \ldots, \pi[i_m]$. Lost packets can make the value of Buf become zero. For example, let $W = 2$ and $\pi = (2\ 3\ 4 \ldots)$; after packet 4 is received packet 1 is considered lost. Consequently 1 is ignored, the buffer is flushed and $Buf_2[3] = 0$. The *reorder buffer-occupancy (RBD_W)* of a permutation of packet IDs π (with threshold W) is defined as the distribution of frequencies of values $Buf_W[i]$.

A subtle point raised by the previous definition concerns the behavior of our equivalences. In the context of RBD_W, where we potentially discard packets, it might be useful to modify the behavior of our equivalence notions to take this fact into account. For instance, if a packet is considered lost (as in the context of RBD_W), it would make sense to increase the value of the corresponding ACK to reflect this fact.

Conclusion 3 *Features of the investigated metric could lead to changes in the definition of the similarity notion.*

Consistent with the above discussion, we will modify the definition of $\equiv_{beh}, \equiv_{buf}, \equiv_{FB}$ to take into account the fact that some packets are deemed lost:

Theorem 6 *The following are true: (i)* RD_W, $W \geq 3$ *is* not *consistent under* $\equiv_{buf}$, $\equiv_{beh}$ *or* $\equiv_{FB}$. *(ii)* RBD_W *is* not *consistent under* $\equiv_{buf}$, $\equiv_{beh}$ *or* $\equiv_{FB}$.

Proof. (i) Sequences of IDs (4 2 3 1) and (4 3 2 1) have different RD distributions but they are buffer, behaviorally, and FB equivalent.

(ii) The first claim is witnessed by sequences (2 3 3 1) and (3 4 1 2), that have identical buffer sequences (1 2 2 0) but different Buf_3 sequences (1 2 0) and (1 2 2 0), respectively.

For the rest, consider first the case $W = 3$.

The second claim is witnessed by sequences (3 3 1 3 3) and (5 3 1 5 3), that have identical ACK sequence (1 1 2 2 4) but different Buf_3 sequences (1 1) and (1 2 1), respectively.

The last claim is witnessed by sequences (5 3 1 5 3 4) and (5 5 1 3 2 4), that have identical FB sequence (5 5 4 4 2 0) but different Buf_3 sequences (1 2 0) and (1 1 2 2 0), respectively.

For a different $W \geq 3$ modify the above examples correspondingly by inserting repeats of packet 3 after the second position, in such a way as to make packet 2 lost in the examples for the second claim and the first one for the last claim.

□

Conclusion 4 *The precise definition of similarity greatly impacts consistency. Different similarity notions can have incomparable sets of consistent metrics.*

5 Consistency: practice meets theory

The consistency criterion in the previous section can be criticized 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 [IHT+06].

One way to address this problem would be to build a set of reordering patterns arising from a "standard set" of real traces and investigate the consistency of the various measures on this set of patterns. This approach has the drawback that it is crucially

dependent on the selection of traces: different sets of real-life traces can lead to different sets of consistent measures. Moreover, such a solution is *not transparent*: in the case when a given measure is consistent with respect to one trace set and inconsistent with respect to another, it is not clear what the reason for this change is.

In the sequel we suggest a different approach: using *reordering invariants* to reduce the set of traces considered in the definition of consistency. The idea is the following: TCP attempts to preserve packet sequence integrity. Therefore measures of reordering of real traffic data fall into three categories:

1. those that, although unbound in theory, have "in practice" values roughly similar to those of a sorted sequence, reflecting the order-preserving nature of TCP, and
2. those that vary significantly both in theory and in practice.
3. those that are small both in theory and in practice. We are not going to be interested in such measures in this paper.

We call measures of the first type *reorder invariants*. Reorder invariants are interesting since they expose structural restrictions on the nature of TCP traces.

The use of reorder invariants provides a transparent way to test consistency: D is a reorder invariant, in that real-life ID sequences X satisfy $D(X) \leq k$ (we will call such a hypothesis *the reorder invariant constraint*) then, to test the inconsistency of M we only test whether there exist Y and Z with $D(X), D(Z) \leq k$, $Y \equiv Z$ and $M(Y) \neq M(Z)$. Suppose measure M, deemed consistent with respect to such a test, was later found to be inconsistent with respect to a different dataset P. Then we can attribute this to a very clear reason: traces in P do not satisfy the reorder invariant constraint.

Clearly a number of issues (e.g. how to perform the test) remain and we do not attempt to solve them here. Instead, we present experimental evidence that a metric called **shuffled up-sequences (SUS)** [ECW92], is a reorder invariant:

Definition 9 The *SUS of a packet sequence* is the minimum number of ascending subsequences into which we can partition the sequence.

SUS itself is an inconsistent measure:

Theorem 7 *SUS is inconsistent with respect to* $\equiv_{beh}$, $\equiv_{buf}$ *and* $\equiv_{FB}$.

Proof. Sequences $A = 4\ 3\ 2\ 1$ and $B = 4\ 2\ 3\ 1$ are congruent with respect to all three similarity measures, yet $SUS(A) = 4$ and $SUS(B) = 3$. □

To provide evidence that SUS is a reorder invariant we will use two sets of traces: one is natural, and the other arises from using the ns-2 traffic simulator. In the latter case we will actively attempt to induce packet reordering in the network. The fact that the SUS measure stays relatively constant even in this "adversarial" framework is evidence for its reorder invariant status.

The set of real packet traces we use was 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

Table 1 Distribution of SUS in real data

SUS	Tr.5 %	Tr.7 %	Tr.8 %	Tr.9 %	Tr.10 %
2	95.07	96.87	98.22	95.29	98.96
3	4.54	2.29	1.66	4.32	0.95
4	0.32	0.16	0.09	0.30	0.07
5	0.03	0.02	0.01	0.05	< 0.01
6	0.01	< 0.01	< 0.01	0.01	< 0.01
≥ 7	—	—	< 0.01	< 0.01	< 0.01
max	9	8	12	16	10

Table 2 Distribution of SUS in ns-2 data (exact to two digits)

Run	2	3	4	5	6	Patterns
1st	> 99.99	< 0.01	0	—	—	38,764
2nd	67.7	27.4	4.6	0.2	< 0.01	37,949

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 [FP99]. Since our aim is to highlight nontrivial network dynamics, we only retain 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 have computed the value of the SUS metric on the network data described above. 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 dumbell topology with two routers and six end-hosts (see Figure 1). 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 unordered states in all three connections, and computed the distribution of SUS values of these sequences. The results are presented in Table 2. Even though the second experiment introduces a significant amount of packet reordering, even in this case more than 50% of sequences 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 having to cope with significant reordering.

One issue raised by the use of reorder invariants is whether the use of such invariants makes the consistency problem interesting: given similarity relation $\equiv$ and reorder invariant D it might be the case that *the only* equivalent sequences $X \equiv Y$ satisfying

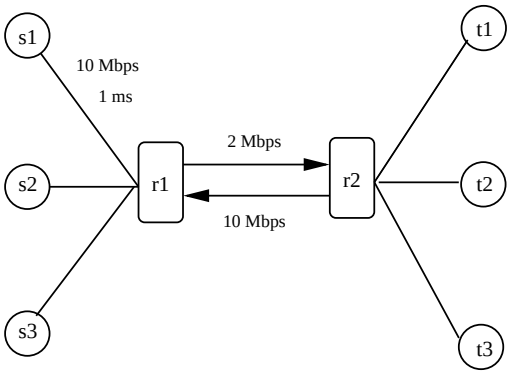

Fig. 1 Simulated network topology.

the conditions $D(X), D(Y) \leq k$ are the (trivial) ones for which $X = Y$. In this case all measures are trivially consistent with respect to sequences satisfying the reordering invariant constraint.

Definition 10 A similarity notion $\equiv$ is *nontrivial with respect to reorder invariant constraint* $D(X) \leq k$ if there exist two sequences $A \neq B$ such that $A \equiv B$ and $D(A), D(B) \leq k$.

Definition 11 A *permutation with repeats* is a sequence of integers $A = A_1, \ldots, A_m$ such that,

1. for some $n \geq 1$,
$$\{A_1, A_2, \ldots, A_m\} = \{1, 2, \ldots, n\}$$
(as sets).
2. For any repeat packet A_i, we have
$$A_i > ACK_i.$$

Example 2 For $D = SUS$ the invariant constraint $D(X) \leq 3$ covers (as seen in Table 1) the vast majority of traces in the UCLA dataset (more than 99.9% of them). The

equivalence measures $\equiv_{beh}$ and $\equiv_{buf}$ are nontrivial with respect to the invariant constraint. This is easily witnessed by sequences $A = (3\ 4\ 1\ 2)$ and $B = (2\ 3\ 3\ 1)$. The situation of measure $\equiv_{FB}$ with respect to the constraint is more complicated, and is dealt with in [Ist08]:

1. First, $\equiv_{FB}$ is trivial for permutations.
2. However, for *permutations with repeats* $\equiv_{FB}$ is nontrivial but well-behaved. In particular in [Ist08] we show how to count all sequences A with $SUS(A) \leq 3$ mapped by FB to a given buffer sequence.

This last example allows us to explicitly provide an example of a reordering measure that is *provably* consistent, under an invariant constraint holding for more than 99% of the traces in Table 1 and Table 2.

Theorem 8 *For permutations with repeats metric RD_W is consistent with respect to the nontrivial similarity notion $\equiv_{FB}$ under the invariant $SUS(A) \leq 3$.*

Proof The proof follows almost immediately from the result in [Ist08]. Indeed, consider two permutations with repeats A and B with $SUS(A), SUS(B) \leq 3$ and $A \equiv_{FB} B$. We need to show $RD_W(A) = RD_W(B)$. Let $\overline{A}$, $\overline{B}$ be the sequences obtained from A and B by eliminating the repeat packets.

Then $\overline{A}, \overline{B}$ are permutations of the same length, $SUS(\overline{A}), SUS(\overline{B}) \leq 3$ and $\overline{A} \equiv_{FB} \overline{B}$ (by the way we extended relation $\equiv_{FB}$ to permutations with repeats. Moreover, elements of $\overline{A}, \overline{B}$ occupy identical positions within sequences A and B.

By applying the result in [Ist08] we infer the fact that $\overline{A} = \overline{B}$ (i.e. the "permutation skeletons" of the two sequences are identical). But notice now that in the definition of measures RD_W *repeat packets don't matter*. This means that we can compute $RD_W(A), RD_W(B)$ by using the (identical) sequences $\overline{A} \equiv \overline{B}$. Thus $RD_W(A) = RD_W(B)$, in other words measure RD_W is consistent. □

6 Conclusions and Acknowledgments

We have introduced a (hopefully interesting) condition, *consistency* that provides a condition any "interesting" metric should satisfy. We have illustrated this condition by investigating the consistency of various metrics to the equivalence notions considered in [HIK06, IH08, IHT+06]. Finally, we have introduced another concept, that of *reorder invariants*, that puts reasonable restrictions on the nature of TCP packet sequence, thus making consistent some real-life metrics.

A natural continuation of this work would be to classify the various reordering metrics from [MCR+06] with respect to their consistency in "real" traces.

This work has been supported by the U.S. Department of Energy under contract W-705-ENG-36, by the Romanian CNCSIS under a PN-II "Idei" grant, and by a grant from the Austrian BMWF.

References

[BBJ02] T. Banka, A. A. Bare, and A. P. Jayasumana. Metrics for degree of reordering in packet sequences. In *Proc. 27th IEEE Conference on Local Computer Networks*, pages 333–342, November 2002.

[BPS99] J. C. R. Bennett, C. Partridge, and N. Shectman. Packet reordering is not pathological network behavior. *IEEE/ACM Transactions on Networking*, 7(6):789–798, December 1999.

[BS02] J. Bellardo and S. Savage. Measuring packet reordering. In *Proc. Internet Measurement Workshop*, pages 97–105, 2002.

[ECW92] V. Estivill-Castro and D. Wood. A survey of adaptive sorting algorithms. *ACM Computing Surveys*, 24(4):441–476, 1992.

[FP99] W. Fang and L. Peterson. Internet-AS traffic patterns and their implications. In *Proc. IEEE GLOBECOM Conf.*, pages 1859–1868, Rio de Janeiro, Brasil, December 1999.

[HIK06] A. Hansson, G. Istrate, and S. Kasiviswanathan. Combinatorics of TCP reordering. *Journal of Combinatorial Optimization*, 12(1–2):57–70, 2006.

[IH08] G. Istrate and A. Hansson. Counting preimages of TCP reordering patterns. *Discrete Applied Mathematics*, 156(17):3187–3193, 2008.

[IHT+06] G. Istrate, A. Hansson, S. Thulasidasan, M. Marathe, and C. Barrett. Semantic compression of TCP traces. In *Proceedings of the IFIP NETWORKING Conference, F. Boavida (editor)*, volume 3976 of *Lecture Notes in Computer Science*, pages 123–135. Springer Verlag, 2006.

[Ist08] G. Istrate. Identifying almost sorted permutations from TCP buffer dynamics. Technical Report 0810.1639, arXiv.org, 2008. (submitted to *Theory of Computing Systems*).

[JPB+08] A. P. Jayasumana, N. M. Piratla, A. A. Bare, T. Banka, and R. Whitner. Improved packet reordering metrics. RFC 5236, 2008.

[KR03] J. Kurose and K. Ross. *Computer Networking: A Top-Down Approach Featuring the Internet, Second Edition*. Addison Wesley, 2003.

[LG02] M. Laor and L. Gendel. The effect of packet reordering in a backbone link on application throughput. *IEEE Network*, 16(5):28–36, September/October 2002.

[MCR+06] A. Morton, L. Ciavattone, G. Ramachandran, S. Shalunov, and J. Perser. Packet reordering metric for ippm. IETF RFC 4737, 2006.

[PJ08] N. Piratla and A. Jayasumana. Metrics for packet reordering - a comparative analysis. *International Journal of Computer Systems*, 21(1):99–113, 2008.

[PJB05] N. M. Piratla, A. P. Jayasumana, and A. A. Bare. RD: A formal, comprehensive metric for packet reordering. In Springer Verlag, editor, *Proc. IFIP Networking Conference 2005*, volume 3462 of *Lecture Notes in Computer Science*, pages 78–89. Springer Verlag, 2005.

[PJBB07] N. Piratla, A. Jayasumana, A. Bare, and T. Banka. Reorder buffer-occupancy density and its applications for measurement and evaluation of packet reordering. *Computer Communications*, 30(9):1980–1993, 2007.

[Rub88] A. Rubinstein. Similarity and decision-making under risk. *Journal of Economic Theory*, 46:145–153, 1988.

[SK83] D. Sankoff and J.B. Kruskal. *Time Warps, String Edits, and Macromolecules: The Theory and Practice of Sequence Comparison*. Addison Wesley, 1983.

[Zad70] L. Zadeh. Similarity relations and fuzzy set ordering. *Information Sciences*, 3:177–200, 1970.