

# Metrics for Degree of Reordering in Packet Sequences<sup>1</sup>

Tarun Banka  
*Department of Computer  
Science*  
tarunb@cs.colostate.edu

Abhijit A. Bare  
*Department of Computer  
Science*  
abhijit@cs.colostate.edu

Anura P. Jayasumana  
*Department of Electrical  
and Computer Engineering*  
anura@enr.colostate.edu

Colorado State University, Fort Collins, CO 80523.

## Abstract

*Out of order arrival of packets is an inevitable phenomenon on the Internet. Application performance can degrade to a great extent due to out-of-order arrival of packets. Metrics to characterize the degree of re-ordering will promote the evaluation of network protocols with respect to packet reordering as well as provide a uniformly defined quantitative measure for packet reordering. We define the concept of “reorder density function” (RD), which quantitatively measures the degree of reordering in sequences of packets. While the reorder density function can provide a comprehensive picture on reordering, measures such as mean and median of reorder density can be used when simpler metrics are desired. Results of reordering analysis using the above metrics for TCP and UDP based applications under different network conditions are presented.*

## 1. Introduction

Out of order arrival of packets is a common phenomenon on the Internet. Major cause of re-ordering has been found to be the inherent local parallelism in switches [2]. Load balancing in network switches introduces local parallelism, which causes packets between same source and destination to take different paths within the switch. Since paths can have different delay characteristics, packets can suffer differential delays and thus, may arrive out-of-order at the destination. Reordering of packets is dependent on two other factors: (i) the configuration of hardware or software in the nodes; and (ii) the load on the intermediate nodes, i.e., switches and routers [2].

Why is it necessary to define metrics to characterize re-ordered sequences of packets? To answer this question, one needs to look at the impact of reordering on applications based on TCP and UDP. In the case of TCP, the impact of reordering depends on the direction in which reordering has taken place, i.e., whether data packets are reordered (forward path reordering) or acknowledgements

are reordered (reverse path reordering). Due to forward path reordering, data packets arrive out of order at the destination, and it has multiple effects on the performance of TCP. When out-of-order packets are received, TCP perceives it as loss of packets, resulting in following [2]:

- (i) As the frequency and amount of reordering increase, the number of unnecessary retransmissions increases and that results in drop in throughput [1,8].
- (ii) The congestion window becomes very small due to multiple fast retransmissions. Thus, TCP has a problem raising the window size, resulting in decreased bandwidth utilization.
- (iii) Due to multiple retransmissions, TCP does not sample the round trip time (RTT) frequently, thus degrading the estimate of RTT.
- (iv) Performance of receiver also suffers, because whenever the receiver receives out-of-order packets, it has to buffer all the out-of-order packets and they need to be sorted as well.
- (v) Detection of loss of packets is delayed because of out-of-order delivery, due to which retransmission request for a lost packet is sent only when TCP times out.

Due to reverse path reordering, when acknowledgements are reordered, TCP loses its self-clocking property, i.e., property of TCP that it only sends packets when another packet is acknowledged doesn't remain valid, resulting in bursty transmissions and possible congestion.

For delay sensitive applications based on UDP, out-of-order arrivals can have a drastic effect on application performance. In case of VoIP, for example, packets need to be received in order and also before playback time, in order to maintain the intelligibility of voice. If an out-of-order packet arrives after its playback time has elapsed, that packet may be treated as lost. It decreases the perceived quality of voice on receiver side. This is only one example of reordering of packets degrading the performance of a UDP application. Other UDP based applications like streaming video also show the same

<sup>1</sup> This research was supported in part by the DARPA Next Generation Internet (NGI) program and the NSF Information Technology Research (ITR) grant no. 0121546.

effect. Reordering can also increase buffer requirements at the receiver as well as processing related latency.

The percentage of out-of-order packets has been used as a metric for characterizing reordering. This method is vague and there is no uniform definition available for this metric. It does not give enough information for characterizing different packet sequences. For example, consider two packet sequences (1,3,4,2,5) and (1,4,3,2,5). It is possible to interpret the out-of-orderness of packets differently in this case, for example,

- (i) Packets 2, 3 and 4 are out-of-order in both cases.
- (ii) Only packet 2 is out-of-order in the first sequence, while packets 3 and 4 are out-of-order in the second.
- (iii) Packets 3 and 4 are out-of-order in both the sequences.
- (iv) Packets 2, 3 and 4 are out-of-order in first sequence, while packets 4 and 2 are out-of-order in the second sequence.

Need for a precise definition is evident from this example. If buffers are available to store packets 3 and 4 while waiting for packet 2, it is possible to recover from the reordering. However, there may be cases where the application requirement is such that arrival of packet 2 after this delay renders it useless. While one can argue that a good packet reordering measurement scheme should capture such effects, a counter argument can also be made that packet reordering should be measured strictly with respect to the order of delivery and should be application independent. These reasons have motivated us to develop metrics to characterize the out-of-orderness of packet sequences.

This paper presents a metric “reorder histogram” (RH), which is a histogram that characterizes the sequences using frequencies of displacement values and reorder density function (RD) its normalized form. Displacement depends on the number of positions in the sequence a packet got delayed. Reorder density can also be viewed as the occupancy distribution of a hypothetical buffer at the receiver end that allows it to recover from out-of-order delivery.

The metric based on the number of SACK blocks developed in [2] is applicable to TCP sessions only. On the other hand, our proposed metrics are applicable to both UDP and TCP traffic on the Internet. IETF Internet drafts available at the moment [4,5,6] propose different types of metrics to measure re-ordering. The acceptable level of reordering depends on the type of application; VoIP and FTP applications, for example, can tolerate different levels of reordering. RH is adaptable to application specific requirements and also captures out-of-order arrival due to loss of packets on the network. All other proposed metrics do not include this option. Metrics in [1,4,5,6] capture re-ordering only due to the late arrival of packets. For example, the scheme presented in this paper

considers the sequence (1,3,5,6,7) as in-order or out-of-order depending upon the displacement threshold value (explained in section 2), but according to the existing metrics, this sequence is considered to be in-order. Note that for on-the-fly characterization of sequence in packet streams, it is not possible to know whether this sequence is in-order or not until one is certain of the fact that packet 2 and packet 4 will not arrive within a useful time period. The RD has provision for including a window after which a packet is considered useless.

The paper also provides results of several experiments carried out to study the effectiveness of the reorder histogram in capturing the degree of reorder in out-of-sequence packets under different network conditions. It includes the evaluation of the effect of router load balancing algorithms on the arrival sequence order of the packets. Besides the reorder density function, the paper demonstrates that several single value metrics like mean, median etc. derived from the reorder density function may be used to characterize reordering in a given packet arrival sequence.

Section 2 describes the metric that we propose for measuring out-of-orderness, i.e. the reorder density (RD). We also provide a computation algorithm with illustrative examples. Section 3 describes several single valued metrics directly derived from the reorder density. Section 4 presents the results of several experiments that use RD for evaluating reordering. Section 5 covers conclusions.

## 2. Reorder Density (RD)

Some important terms are first defined that will help us describe the reorder histogram (RH) and its normalized form, the reorder density function (RD). Then, an algorithm for computing the histogram and density is presented, followed by some illustrative examples.

### 2.1. Definitions

- Displacement (D): Displacement is the number of positions in the sequence the expected packet got delayed. At every arrival of a packet, we can compute the displacement value depending upon the expected sequence number. For example, for the sequence of packets (1,2,4,5,3), the displacement value of packet 3 is 2 since it has arrived 5<sup>th</sup> in the sequence, while it was expected to be 3<sup>rd</sup> in the sequence. The value of displacement of packet 3 becomes 1 when packet 4 arrives instead of packet 3 and it becomes 2 when packet 5 arrives.
- Displacement Threshold ( $D_t$ ): This parameter can be defined as the tolerance of the application to the maximum allowed displacement of a packet. Any

packet that gets displaced by more than the value of this threshold will be considered as a lost packet. The threshold is such that even if the packet ultimately arrives after threshold, it is of no use for the application. Many factors, influence the selection of the displacement threshold value, for example, the transport layer protocol (UDP or TCP), the amount of redundant information sent to recover from losses, and whether the sequence of packets belong to a time-sensitive application or not. In case of a VoIP application, with a bit-rate of 128 kbps and a packet size of 200 bytes,  $D_T$  value can be determined as follows. Assume that the application can wait maximum 50 ms for expected packet, and that the packets arrive at a constant rate. That means within 50 ms, the application can receive  $(128 * 1000 * 0.05) / (200 * 8)$  i.e. 4 packets. Therefore, the displacement threshold should be kept at 4. In case of TCP, a lost or delayed packet will be retransmitted and reach the destination. So the value of  $D_T$  should be kept at least equal to the size of receiving window on receiver side.

- **Buffer Occupancy:** An arrived packet with a higher sequence number than that is expected may be stored in a buffer. At any point of time, the buffer occupancy is the number of such out-of-order packets (assuming one buffer for each packet). The value of the buffer occupancy is actually the value of displacement at that point of time. Note that the value of buffer occupancy is not the memory buffer provided by the operating system of the host, rather it is the size of a hypothetical buffer that would be required if all the out-of-order packets are to be buffered.
- **Reorder Density (RD):** The reorder histogram is defined as the histogram of occupancy of the hypothetical buffer. Reorder histogram (RH) consists of the array  $F[i]$ ,  $i=0, \dots, D_T$ , where  $F[i]$  is the number of arrival instances, at which  $i$  buffers were occupied. Normalization of this array with respect to  $\sum F[i]$  provides the reorder density function (RD). RD facilitates the comparison of different sequences independent of their length. In the following discussion, we use the two terms RD and RH interchangeably.

## 2.2. Computation of Reorder Density (RD)

Let  $N$  be the total number of packets in the sequence, and  $S$  be the sequence number of an arrived packet. There may be missing or duplicate sequence numbers in the arrival sequence. In case of duplicate packet sequence numbers, we take into account only the first arrival of that packet. Let  $E$  be the expected sequence number. When the expected packet is received,  $E$  is incremented by 1 (i.e. next packet in sequence is now expected). For simplicity,

```

# E : next expected sequence number.
# S : sequence number of packet just arrived.
# D : estimated displacement of the arrived packet from
its expected position in sequence.
#  $D_T$  : displacement threshold.
#  $F[i]$  : Frequency of occurrence of  $D = i$ .
# received(E) : true if packet with sequence number E is
already received, false otherwise. A packet just arrived is
also considered to be received.

1. Initialize  $E = 1, D = 0$  and  $F[i] = 0$  for all values of  $i$ 
( $0 \leq i \leq D_T$ ).
2. Repeat the following for each arrived packet with
sequence number S.
(a) if (received(E));
    # This indicates duplicate packet. Do nothing.
(b) if ( $S == E$ ) {
    # Arrived packet is the expected one. Remove
this and any other in-sequence packets from buffer.
     $E = E + 1$ ; # Expect next packet
    while (received(E)) {
         $D = D - 1$ ;
         $E = E + 1$ ;
    }
    # Update frequency for displacement D.
     $F[D] = F[D] + 1$ ;
}
(c) if ( $S > E$ ) {
    # Arrived packet is a packet with a higher
sequence number than the expected one.
    if ( $D < D_T$ )
        #Increment Displacement D
         $D = D + 1$ ;
    else {
        # Expected packet is delayed beyond  $D_T$ .
        # Treat it as lost.
        while (not received(E))
             $E = E + 1$ ; # Expect next packet
        # Packet with sequence number E is already
received. Expect next packet.
         $E = E + 1$ ;
        while (received(E)) {
             $D = D - 1$ ;
             $E = E + 1$ ;
        }
    }
    # Update frequency for displacement D
     $F[D] = F[D] + 1$ ;
}

```

**Figure 1. Reorder Density (RD) Computation Algorithm**

this discussion assumes that the sequence numbers of packets transmitted are consecutive integers. However, the RH/RD is also applicable for cases where the sequence number is incremented by any value. For example, with TCP,  $E$  is incremented by the size of the payload in the

packet, but it does not affect the generality of the approach. For examples discussed in the paper, the initial value of E is assumed to be 1, whereas in more general cases, E is the first expected sequence number.

The algorithm for computing the reorder histogram (RH) is shown in figure 1. The algorithm starts with initialization of D to 0 and E to 1. If an arrived packet with sequence number S is the expected packet (part 2.b), E is incremented by 1 (i.e. next packet in sequence is now expected). If packet with new E, i.e., the next packet in sequence has already arrived, it need not be stored in the buffer any more (it can be used by the application). So displacement value is reduced by 1 and E is incremented by 1. This is repeated till all the waiting packets that are in sequence are removed.

If the received packet with sequence number S is not the expected packet, two cases arise. First case is when S is higher than E (part 2.c), i.e., received packet is an out-of-order packet. If the displacement is less than the displacement threshold, the packet with sequence number E can still be expected. The value of displacement is incremented, because newly arrived packet needs to be stored in the hypothetical buffer. On the other hand, if displacement is equal to the displacement threshold, the currently expected packet E is assumed to be lost and E is incremented repeatedly till E reaches the sequence number of a packet already received. This packet can now be removed from the hypothetical buffer giving space to newly arrived packet. E is incremented further to check if there are any packets with higher sequence numbers already arrived and waiting. This is similar to the first case. The remaining case, with  $S < E$  means that S is a duplicated or packet delayed beyond  $D_T$ , and hence it is simply ignored (part 2.a). Frequency value for new value of displacement is incremented as shown in the algorithm.

Once the algorithm deals with all the packets, we have computed the frequency  $F[D]$ , for all D, and hence RH.

### 2.3. Examples of reorder density function (RD)

**2.3.1. Case of no packet loss.** Consider the sequence of packets (1,2,3,5,6,7,4,9,8,10,12,13,14,15,11). Let  $D_T$  be 4. Here all the packets of a sequence have been received without loss or duplication. Table 1 shows steps in computing displacements for RH. Initially, (see column 1) E, the expected sequence number is 1 and S, the arrived sequence number is also 1. The displacement value thus is 0 and hence frequency corresponding to displacement=0,  $F[0]$  becomes 1. For next two arrivals, S is equal to the expected sequence number. The value of displacement remains 0 and  $F[0]$  becomes 2 and then 3. At fourth arrival however  $S=5$  is not equal to the expected sequence number  $E=4$ , and hence the displacement value D, is increased by 1. Now frequency of displacement is 1, and  $F[1]$  becomes 1. When the next packet  $S=6$  comes in, E is 4. Displacement value, D, is incremented, i.e.  $D=2$  as one more packet needs to be buffered. Thus,  $F[2]$  is incremented. The process repeats for each arriving packet. Table 2 shows these computed frequency values. Frequency for  $D > 4$  is always 0 as displacement threshold is 4.

Figure 2 shows the reorder density function for above sequence. The absolute frequency for displacement 0 is 7 indicating that there were seven arrival instances when the buffer requirement was 0. The normalized value is  $7/15$  i.e. 0.4667. Similarly, displacement of 3 has a frequency of 2. That means a buffer of size 3 was required during two instances to store out-of-order packets. Highest displacement of 4 means that with a buffer size of 4, the

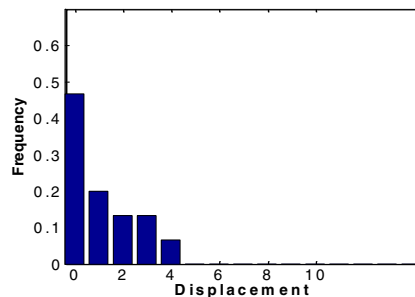
**Table 1. RH Frequency computation for sequence with no packet loss**

|                         |          |          |          |          |          |          |          |          |          |          |          |          |          |          |          |
|-------------------------|----------|----------|----------|----------|----------|----------|----------|----------|----------|----------|----------|----------|----------|----------|----------|
| E                       | 1        | 2        | 3        | 4        | 4        | 4        | 4        | 8        | 8        | 10       | 11       | 11       | 11       | 11       | 11       |
| S                       | 1        | 2        | 3        | 5        | 6        | 7        | 4        | 9        | 8        | 10       | 12       | 13       | 14       | 15       | 11       |
| D                       | 0        | 0        | 0        | 1        | 2        | 3        | 0        | 1        | 0        | 0        | 1        | 2        | 3        | 4        | 0        |
| Frequency<br>( $F[D]$ ) | $F[0]=1$ | $F[0]=2$ | $F[0]=3$ | $F[1]=1$ | $F[2]=1$ | $F[3]=1$ | $F[0]=4$ | $F[1]=2$ | $F[0]=5$ | $F[0]=6$ | $F[1]=3$ | $F[2]=2$ | $F[3]=2$ | $F[4]=1$ | $F[0]=7$ |

E: next expected sequence number. S: sequence number of packet just arrived. D: displacement.

**Table 2. RH/RD corresponding to Table 1**

| Displacement (D) | Frequency $F[D]$ | Normalized Frequency |
|------------------|------------------|----------------------|
| 0                | 7                | 0.4667               |
| 1                | 3                | 0.2                  |
| 2                | 2                | 0.1333               |
| 3                | 2                | 0.1333               |
| 4                | 1                | 0.0667               |
| 5                | 0                | 0                    |
| 6                | 0                | 0                    |



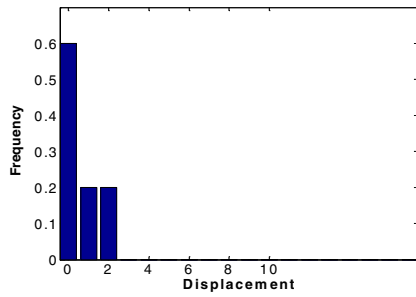
**Figure 2. RD for table 1 ( $D_T = 4$ )**

**Table 3. RH Frequency computation for sequence with loss of packets**

|        |   |   |   |   |   |   |   |    |    |    |    |    |    |    |    |
|--------|---|---|---|---|---|---|---|----|----|----|----|----|----|----|----|
| E      | 1 | 2 | 3 | 4 | 4 | 7 | 8 | 8  | 11 | 11 | 14 | 15 | 16 | 17 | 18 |
| S      | 1 | 2 | 3 | 5 | 6 | 7 | 9 | 10 | 12 | 13 | 14 | 15 | 16 | 17 | 18 |
| D      | 0 | 0 | 0 | 1 | 2 | 0 | 1 | 2  | 1  | 2  | 0  | 0  | 0  | 0  | 0  |
| F[ D ] | 1 | 2 | 3 | 1 | 1 | 4 | 2 | 2  | 3  | 3  | 5  | 6  | 7  | 8  | 9  |

receiver can recover from reordering.  $D_T$  value selected is 4, which implies that any packet arriving with displacement  $D > 4$  is discarded.

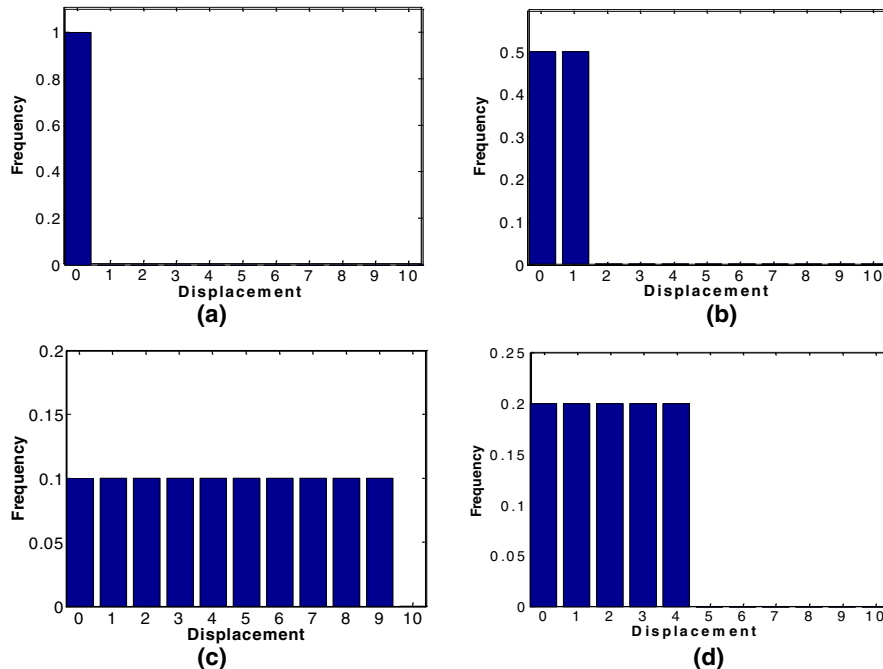
**2.3.2. Case of loss of packets.** A certain sequence number may not be present in a given packet arrival sequence at the time it is expected because the packet was either lost in the network or it was delayed. RH expects such a sequence number until displacement  $D \leq D_T$ , but when  $D$  exceeds  $D_T$ , RH assumes the packet to be lost and proceeds.



**Figure 3. RD for table 3 ( $D_T = 2$ )**

Consider the sequence (1,2,3,5,6,7,9,10,12,13,14,15,16,17,18). Here, packet 4 is either lost or delayed. Let  $D_T$  be 2. Table 3 shows steps in computing displacements for RH. As we observe the displacement value is never allowed to exceed the displacement threshold  $D_T$ . At the end of the sequence,  $F[0]=9$ ,  $F[1]=3$ , and  $F[2]=3$  resulting in the distribution of normalized frequency versus displacement shown in figure 3. When packets are delayed by more than the displacement threshold, the effect is the same as with lost packets.

**2.3.3. Other examples of RD.** Figure 4 shows a few more RD plots for a sequence of 10 packets and a  $D_T \geq 10$ . RD plot of figure 4 (a) shows the scenario of a sequence with no reordering (1,2,3,4,5,6,7,8,9,10). When all packets arrive in sequential order, displacement 0 has normalized frequency of 1.0 and all other displacement values have normalized frequency of 0. This is always the case when no reordering is present in the sequence. Figure 4(b) is the case when every pair of packets (like 1 and 2, 3 and 4 etc.) is reversed resulting in sequence (2,1,4,3,6,5,8,7,10,9). Figure 4(c) illustrates the case of complete reversal of the sequence i.e. (10,9,8,7,6,5,4,3,2,1). For a completely



**Figure 4. Reorder density function (RD) (a) for in-order sequential arrival (b) every pair of packets reversed (c) completely reversed sequence (d) every fifth packet is delayed by five units. ( $D_T \geq 10$ )**

reversed sequence of packets, displacement of an arrived sequence number  $S$  ( $1 \leq S \leq N$ ) is given by  $(N - S)$ , where  $N$  is the maximum sequence number in the arrival sequence. Hence the absolute frequency of all displacement values (i.e. 0 to  $N - 1$ ) is 1 and normalized frequency is  $1/N$  as seen in figure 4(c). Figure 4(d) is the RD plot for sequence (2,3,4,5,1,7,8,9,10,6).

In figure 4(b) the maximum displacement is 1 indicating that with a buffer size of 1, one can recover completely from out-of-order packets. The arrival sequence of figure 4(d) requires 4 buffers to recover completely from all out-of-order packets. From the above example, it can be seen that the reorder histogram provides a detailed view of an arrival sequence. Such a characterization cannot be obtained by a simple metric such as the percentage of reordered packets. The histogram can also be used to determine the number of buffers required to store out-of-order packets for a certain percentage loss as well as to compare the impact on reordering due to protocols like UDP, TCP and different load balancing algorithms, used by network routers.

### 3. Simple metrics derived from RD

While the reorder density can provide a detailed picture of the out-of-orderness present in a sequence of packets, there may be instances, where a simpler metric is needed to compare sequences. Parameters derived from the reorder density that can act as metrics for packet reordering are:

- 90<sup>th</sup> percentile: This parameter is the displacement value, such that 90 % of the arrived packets have displacement less than this value.
- Median: Half of the arrived packets have displacements less than the median and the rest have displacement value greater than the median.
- Mean: It is the average displacement of arrived packets, as given by the mean of RD.

## 4. Results

We performed several experiments to study the effectiveness of RD in capturing packet disorder. We also demonstrate the effects on reordering due to load balancing in routers, and variable delays and losses suffered by UDP and TCP packets on the network. In the following section, we present a description of these experiments along with important results.

### 4.1. Effect of load balancing algorithms on reordering

To study the effects of load balancing on packet reordering, and to evaluate the effectiveness of RD in capturing reorder effects, we simulated a network router, which implements Cisco CEF [9] switching. There are two types of CEF switching, per-packet and per-destination. CEF per-destination based load balancing allows the router to use multiple paths to achieve load sharing. In this case, packets for a given source-destination host pair always take the same path, even if multiple paths are available. In this case, reordering is possible only if the subsequent nodes reorder the packets. In per-packet based load balancing, a round robin scheme is used to determine the path taken by packets to reach the destination. In this case, out-of-order arrival of packets is more likely.

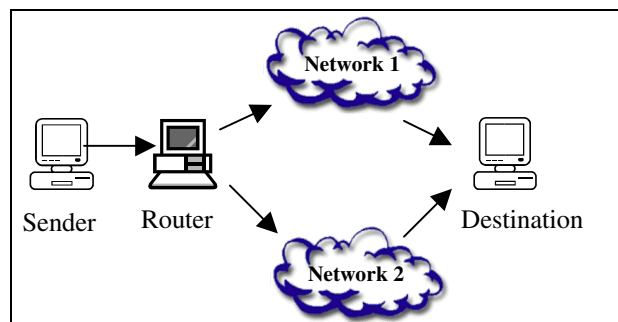
For this experiment we have used actual delay traces for links, collected using ping utility for geographically distributed hosts. Table 4 shows the one-way mean and standard deviations of the delay distributions.

**Table 4. One-way mean and standard deviation values of delay distributions used in experiments**

|       | Destination       | One way mean delay in ms ( $\tau$ ) | Standard deviation in ms ( $\sigma$ ) |
|-------|-------------------|-------------------------------------|---------------------------------------|
| Net-1 | mit.edu           | 19.97                               | 0.20                                  |
| Net-2 | www.iitb.ernet.in | 150.19                              | 11.43                                 |
| Net-3 | jin.jcic.or.jp    | 98.77                               | 3.47                                  |

*All readings from source host idefix.engr.colostate.edu on May 11,2002,20 :00 MST.*

**4.1.1. Results of CEF per-packet load-balancing algorithm for UDP data.** Figure 5 depicts the setup used for this experiment. The packet delay across networks 1 and 2 were independently distributed using delay traces given in table 4.



**Figure 5. Simulation setup for per-packet load balancing**

Figure 6 shows the reorder histograms of simulation results for this experiment under different link bandwidth conditions. For simplicity, both outgoing links of router are assumed to have the same bandwidth.

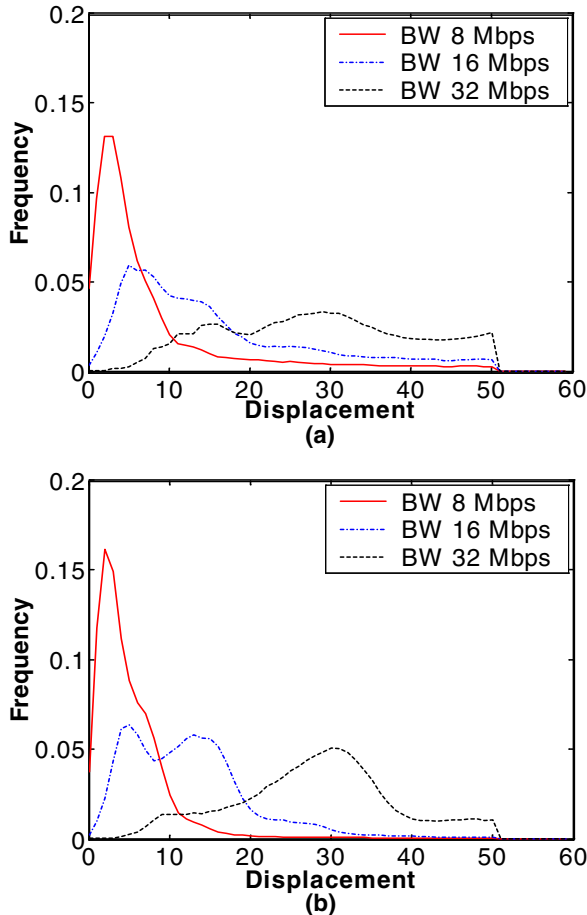


Figure 6. RD for per-packet load balancing using delay distribution for (a) Net-2 ( $\tau = 150.19$  ms,  $\sigma = 12.43$  ms) and (b) Net-3 ( $\tau = 98.77$  ms,  $\sigma = 3.47$  ms). ( $D_r = 50$ )

As seen in figure 6, for higher bandwidth links, larger the displacement value, higher is the frequency. The reason for this behavior is that as the bandwidth of the links increases, more packets go out on the link. In other words degree of re-ordering increases as link bandwidth increases because larger buffers are used more frequently with increase in bandwidth. That increases the parallelism between the transmissions of packets through the network. This results in more reordering.

Table 5 shows the 90<sup>th</sup> percentiles for reorder density function in figure 6. It demonstrates that 90<sup>th</sup> percentile is effective in distinguishing the reorder characteristics when a simple metrics is required.

Table 5. 90<sup>th</sup> percentile level values of RD

| Bandwidth (Mbps) | Figure 6(a) | Figure 6(b) |
|------------------|-------------|-------------|
| 8                | 21          | 8           |
| 16               | 34          | 21          |
| 32               | 44          | 40          |

**4.1.2. Results of CEF destination-based load balancing algorithm for UDP data.** In destination-based load balancing, the router decides the outgoing link to send the packet, based upon the address of the destination of the packet. So all packets for a particular source-destination will always go on a certain outgoing link. For this, we simulated a router with only one outgoing link. Figure 7 shows the simulation setup. In case of per-packet load balancing, reordering may still be introduced in the network through which packets travel after leaving the router. Delay values given in table 4 have been used to simulate network characteristics. Figure 8 shows the RD for same two cases as in figure 6. Comparing figures 6 and 8, RD in figure 8 are seen to be concentrated towards lower displacements. This indicates less packet reordering with destination-based routing compared to per-packet load balancing.

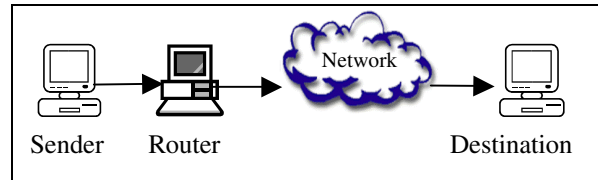
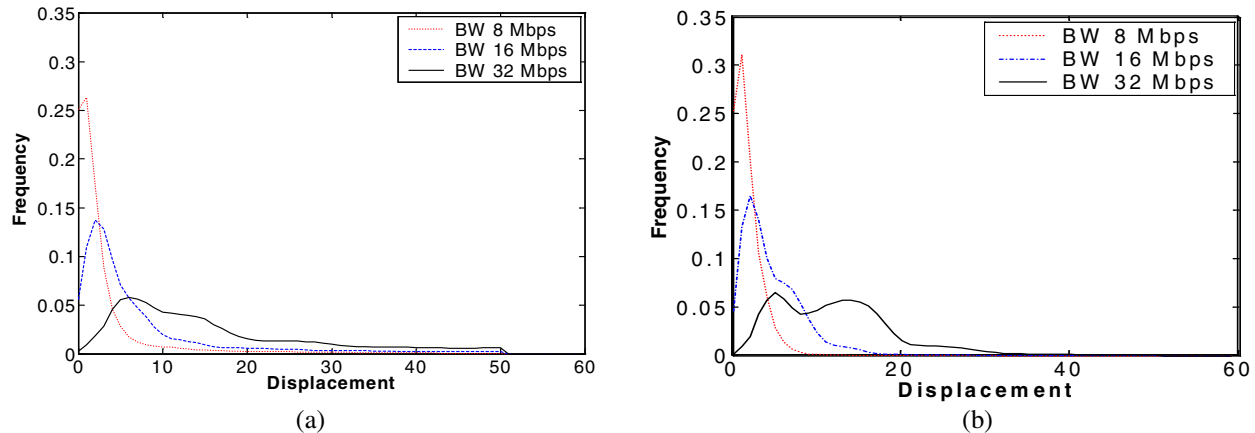


Figure 7. Simulation setup for destination-based load balancing

#### 4.2. Effect of network traffic conditions on reordering for UDP streams

NISTNet [10], a network emulator, was used to emulate different network traffic conditions. We used a UDP based file transfer client-server program. The packets travel through the network emulated by NISTNet. This network introduces packet loss and one-way mean delay from server to client. The payload in each UDP datagram carried a sequence number and data to be transferred. Sequence number was used to detect out-of-order packets. Figure 9 depicts the experimental setup. Figure 10 shows reorder density function (RD) under different delay and packet loss conditions. Considering reorder density in figure10 (a)-(c) (wherein packet loss is varied, keeping delay constant), as the loss rate increases higher displacement values have higher frequency. The reason for this is that in case of UDP, receiver side can never receive a packet once lost, as there are no



**Figure 8. RD for destination-based routing using delay distribution for (a) Net-2 ( $\tau = 150.19$  ms,  $\sigma = 12.43$  ms) and (b) Net-3 ( $\tau = 98.77$  ms,  $\sigma = 3.47$  ms). ( $D_r = 50$ )**

retransmissions. The displacement value keeps increasing till the displacement threshold value is reached, or in other words, subsequent packets have to be buffered waiting for this packet.



**Figure 9. Experimental setup used for UDP and TCP**

Considering figure 10(d) (wherein delay is varied, keeping the loss rate constant), it seems that for constant loss, as one-way mean delay increases, the frequency of higher displacement values also increases. But it has been found with further experiments that increase in standard deviation in mean delay is the reason behind the increase in frequency of higher displacement value. Figure 11 shows the results of these experiments. When the mean delay is kept constant and the standard deviation in mean delay is increased, a high degree of reordering is introduced, as is evident from the results shown in figure 11(a). The reason for this behavior is that with the increase in the value of standard deviation, packets for the same traffic stream suffer larger variations in delays thus increasing the degree of re-ordering. Also it is found that when mean delay is increased keeping the standard deviation constant, no significant changes in reordering are visible. This is evident from the results shown in figure 11(b).

It is observed from the delay characteristics of Internet that as the mean delay increases the corresponding standard deviation in delay also increases. That explains the results shown in figure 10(d). Same explanation applies to figure 6 and 9 as well.

To get an idea of how other metrics specified in this paper change with network traffic parameters, we provide table 6 for the UDP stream with one-way mean delay 150 ms and standard deviation 12 ms corresponding to reorder density function in figure 10 (a).

All the metrics demonstrate the fact that reordering in the sequence increases with packet loss, and these single value metrics are effective in characterizing reordering.

**Table 6. Other metrics for UDP stream of figure 10 (a)**

| Packet Loss (%) | 90 <sup>th</sup> Percentile | Median | Mean  |
|-----------------|-----------------------------|--------|-------|
| 1               | 47                          | 39     | 38.81 |
| 5               | 48                          | 42     | 41.30 |
| 10              | 49                          | 45     | 43.93 |
| 20              | 49                          | 46     | 46.12 |

### 4.3. Effect of network traffic conditions on reordering for TCP streams

Next we study the effect of network conditions on the reordering in TCP packet streams. As mentioned in section 2.1, for TCP, the value of displacement threshold should be kept sufficiently high, to take into account retransmissions. The receiver will always receive a packet with a given sequence number even though it is lost or delayed. Experimental setup is shown in figure 9. An FTP application (TCP based) was used to send the packet from server to client. Figure 12 shows the reorder histograms under different network traffic conditions. In figure 12(a) (wherein the packet loss is varied, keeping the mean delay constant), increasing loss rate does not have much effect on the (RD). Comparing impact of network conditions on UDP and TCP from figure 10 and 12, the displacement

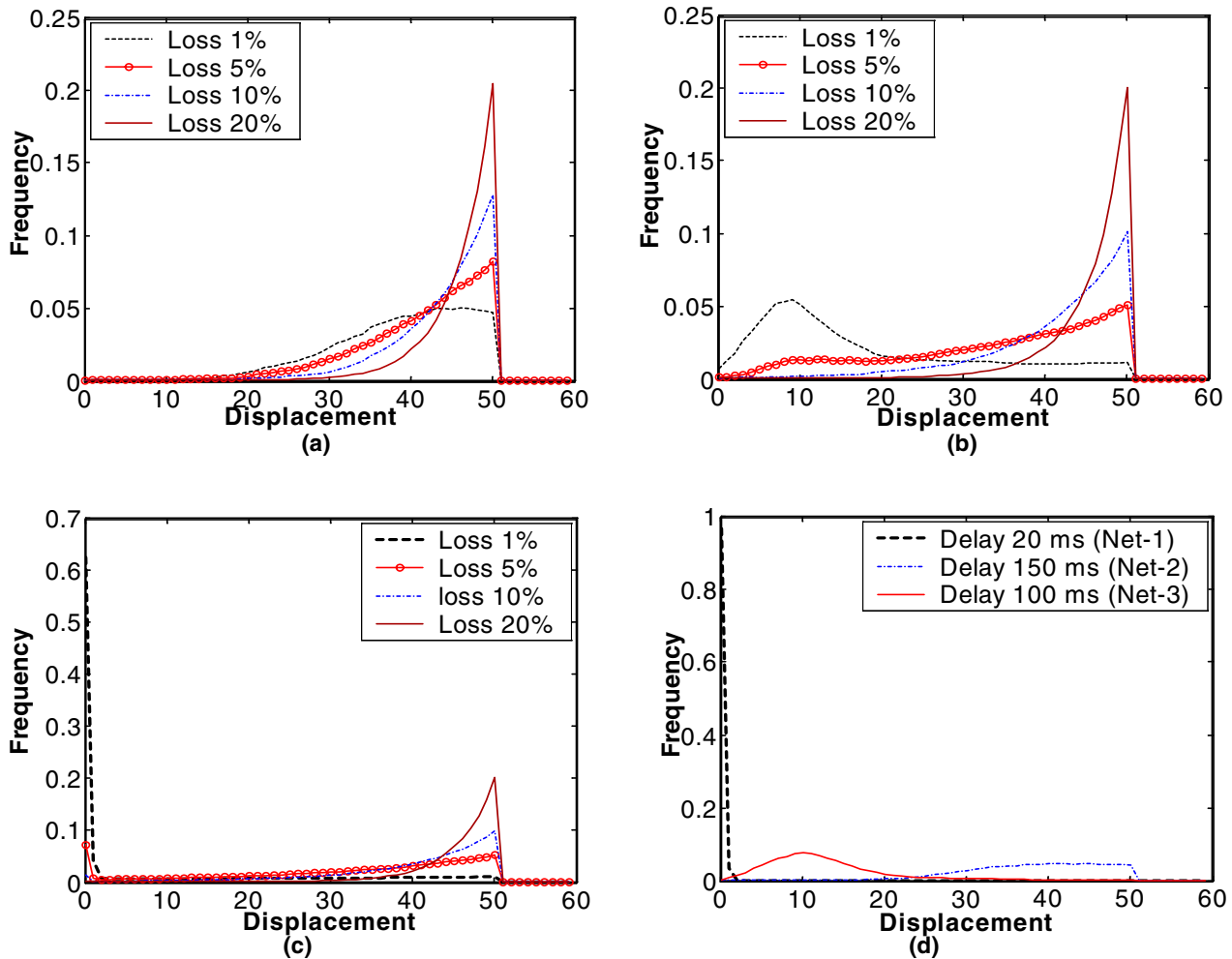


Figure 10. RD for UDP for network traffic conditions (a) Net-1 (b) Net-2 (c) Net-3 and (d) No packet loss. ( $D_\tau = 50$ ) (Net-1,2,3 represent  $\tau$  and  $\sigma$  values according to table 4).

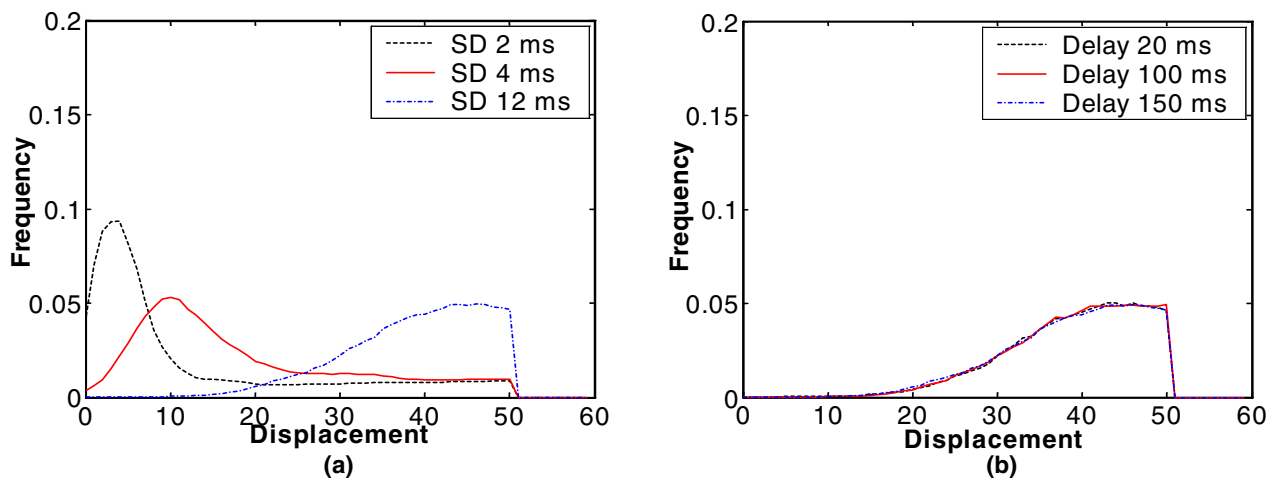


Figure 11. RD for UDP. (a)  $\tau = 150$  ms and packet loss rate 1%. (b)  $\sigma = 12$  ms and packet loss rate 1%. ( $D_\tau = 50$ )

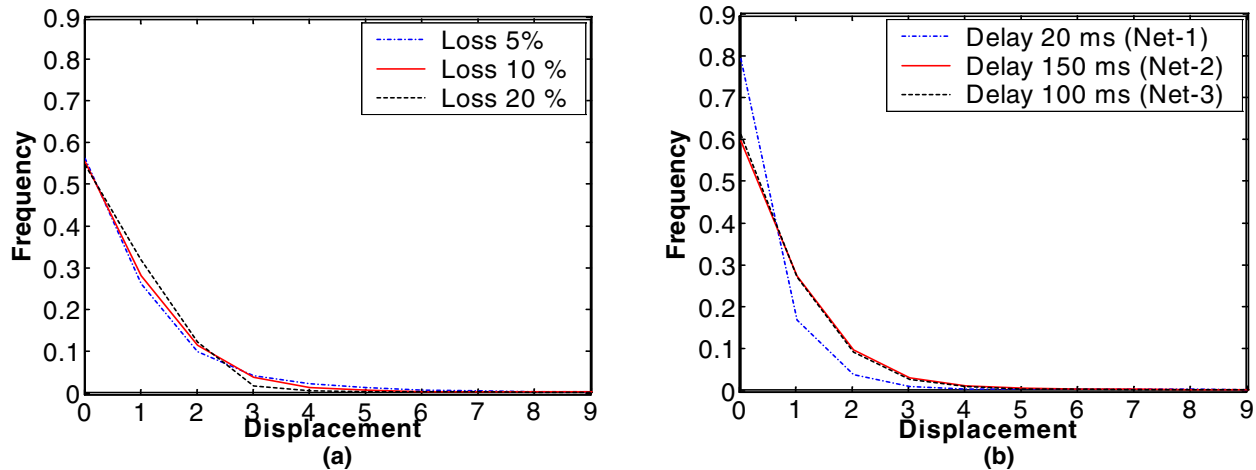


Figure 12. RD for TCP. (a)  $\tau = 150$  ms,  $\sigma = 12$  ms (b) No packet loss ( $D_r = 65536$ )

values are much smaller in TCP, indicating lesser amount of reordering, than that in UDP. This happens because of TCP's slow start algorithm. Whenever packets are lost, TCP reduces the rate at which packets are sent, thus decreasing the outstanding packets and hence, out-of-order arrival of packets. In figure 12(b) (wherein delay is varied, keeping the loss rate constant), with increase in one-way mean delay, the frequency of higher displacement increases, indicating higher degree of reordering. However, as explained in section 4.2, increase in standard deviation in mean delay is the main reason for the increase in reordering.

## 5. Conclusions

This paper introduced a new metric, reorder density function (RD), to represent the reordering of packets in a stream. Conventional packet reorder metrics are quite vague and provide little information about the degree of reordering. Reorder histogram is a frequency distribution against the displacement of packet from its expected position in the sequence. It can also be viewed as the occupancy distribution of a hypothetical buffer at the receiver end required to recover from the out-of-order sequencing. Thus, it is also useful in computing the buffer requirement for a certain packet stream. It is capable of giving comprehensive information on the degree of reordering observed in the arrival sequence. It is a complete histogram or a density function. Thus it is possible to use its mean, median, 90<sup>th</sup> percentile, the standard deviation or a combination of these for expressing the reordering when necessary.

RD provides a formal definition for evaluating reordering introduced by networks and network nodes. It can be used to determine the buffer allocation at intermediate as well as destination nodes. Transport

protocols, for example TCP, with acknowledgement generation and retransmission rules based on RD of the underlying network, have the potential of providing enhanced performance. RD may be used for network diagnosis purposes as well. Abrupt changes in the RD function may be an indication of abnormal network conditions. RD may also be used to evaluate the effects of routing and load balancing algorithms on reordering.

## References

- [1] M. Allman and E. Blanton. "On Making TCP More Robust to Packet Reordering", ACM Computer Communication Review, 32(1), January 2002.
- [2] C. R. Bennett, C. Patridge and N. Shectman. "Packet Reordering is Not Pathological Network Behavior", Trans. on Networking IEEE/ACM, December 1999.
- [3] L. Ciavattone and A. Morton. "Out-of-Sequence Packet Parameter Definition (for Y.1540)", Contribution number T1A1.3/2000-047, October 30<sup>th</sup>, 2000. <ftp://ftp.t1.org/pub/t1a1/2000-A13/0a130470.doc>
- [4] A. Morton, L. Ciavattone and G. Ramachandran. "draft-morton-ippm-nonrev-reordering-00.txt", IETF, work in progress.
- [5] V. Paxson, G. Almes, J. Mahdavi and M. Mathis. "Framework for IP Performance Metrics", RFC 2330, May 1998.
- [6] D. Pullin, A. Corlett, S. Critchley and B. Mandeville. "draft-critchley-mlas-reordering-00.txt", IETF, work in progress.
- [7] S. Shalunov. "draft-shalunov-reordering-definition-00.txt", IETF, work in progress.
- [8] W.R Stevens. "TCP Slow Start, Congestion Avoidance, Fast Retransmit and Fast recovery algorithm", RFC 2001, January 1997.
- [9] Load Balancing with Cisco Express Forwarding: [http://www.cisco.com/warp/public/cc/pd/ifaa/pa/much/prodlit/loadb\\_an.htm](http://www.cisco.com/warp/public/cc/pd/ifaa/pa/much/prodlit/loadb_an.htm).
- [10] NISTNet network emulator. <http://snad.nsl.nist.gov/itg/nistnet>.