Internet Engineering Task Force
INTERNET-DRAFT
draft-jayasumana-reorder-density-00.txt

Anura Jayasumana
Nischal M. Piratla
Abhijit A. Bare
Tarun Banka
Colorado State University

February 2003
Expires: August 2003

Reorder Density Function - Metric for packet reordering measurement

Status of this memo

Abstract

Out of order arrival of packets can degrade the performance of many TCP-based, VoIP-based and Video-based applications significantly. There is a need for a metric that can meaningfully, accurately and unambiguously characterize reordering. This memo proposes a new metric called Reorder Density function (RD), which can give an in-depth view of the reordering present in any packet sequence. This well-defined metric can also be used to evaluate effects of protocol and hardware implementations on packet reordering. The memo also provides an algorithm to compute the reorder density function followed by some illustrative examples.

1. Introduction and Motivation

Out of order arrival of packets is a common phenomenon on the Internet. Major cause of reordering of packets is the local parallelism present in network routers and switches. This parallelism is caused due to different load balancing algorithms used in routers and switches. Packets can also be reordered due to different queuing schemes within the networking equipment itself. The reordering leads to degradation of the performance of the applications. For example, perceived quality of voice degrades if VoIP application receives packets out of order. Once we are able to quantify the degree of reordering in arriving packet streams, it is possible to predict the effects of reordering on applications that are sensitive to reordering, and perhaps even compensate for reordering. This can further help us in evaluating network protocols with respect to packet reordering.

Until now, the percentage of out of order packets has been used as a metric for characterizing reordering. However, this metric is vague and lacks in detail. There is no uniform definition of the out-of-orderness of an arrived packet. 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-orderedness 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 2 and 3 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.

In essence, the percentage of out-of-order packets is subject to interpretation and hence it cannot capture the out-of-orderness unambiguously and, hence, accurately. Thus, the current definition is not a sufficient metric and there is a need for a more precise and a complete definition.

Taking any of the above sequences, 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.

In this memo, we define a metric called Reorder Density Function (RD), which is a normalized reorder histogram that characterizes packet sequences using frequencies of displacement. Displacement depends on the number of positions in the sequence a packet got delayed. Thus, the displacement accounts for the distance an expected

packet has moved relative to its original position. RD can provide not only a consistent percentage out-of-orderedness measure, but also can be used to compute the percentages corresponding to different degrees of out-of-orderedness. 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. Next sections explain the concept of the reorder density function (RD).

2. Definitions of terms used

Some important terms are defined which will help us describe the Reorder Density Function (RD) algorithm.

2.1 Out of order packet :

When a packet other than the expected packet arrives, it is considered as an out of order packet, provided it is not a duplicate of an already received packet.

2.2 Displacement (D):

Displacement is the number of positions in the sequence an 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 of packet with sequence number 3 is 1, when the packet with sequence number 4 arrives, and it increases to 2, when the packet with sequence number 5 arrives.

2.3 Displacement Threshold (DT):

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 to be a lost packet. The threshold is chosen such that even if the packet ultimately arrives after threshold, it is of no use to 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 VoIP application, for example, with a bit-rate of 128 kbps and packet size of 200 bytes, DT value can be determined as follows. Assume that the application can wait maximum 50 ms for expected packet, and that the packets arrive at 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 will reach the destination. So the value of the DT should be kept at least equal to the size of receiving window on receiver side.

2.4 Buffer Occupancy:

Whenever a packet with sequence number greater than the expected sequence number arrives, it is considered to be stored in a buffer. This is a hypothetical buffer used for calculating RD and it does not mean that the packet is actually stored. At any point of time, the buffer occupancy is the number of out of order packets (assuming one buffer per packet). The reorder density (RD) is the normalized histogram of the buffer occupancy.

3. Algorithm to compute reorder density (RD) function

This section describes an algorithm to compute the reorder density function. Without loss of generality, the description assumes that the sequence numbers start at 1 and increment by 1 for each packet.

```
-------------------------------------------------------------------
# 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.
# DT : Displacement threshold. (DT is also the size of the buffer
  allocated.)
# F[i] : Frequency of occurrence of D = i.
# RD[i] : Normalized frequency for D = i.
# in_buffer(N) : True if the packet with sequence number N is
  already stored in the buffer.
===================================================================
```

1. Initialize E = 1,D = 0 and F[i] = 0 for all values of i (0 <= i <= DT).

2. Do the following for each arrived packet.

```
If (in_buffer(S) || S < E) /*Do nothing*/;
/*Case a: S is a duplicate or delayed packet. Discard the
  packet.*/

ElseIf (S == E)
/*Case b: Expected packet has arrived.*/
{
   E = E + 1;
   While (in_buffer(E))
   {
      D = D - 1; /*Free buffer occupied by E.*/
```

```
            E = E + 1; /*Expect next packet.*/
          }
        } /*End of ElseIf (S == E)*/
```

```
        ElseIf (S > E)
        /*Case c: Arrived packet has a sequence number higher
          than expected.*/
        {
           If (D < DT)
           /*Store arrived packet in buffer.*/
              D = D + 1;
           Else
           /*Expected packet is delayed beyond DT. Treat it as
             lost.*/
           {
               Repeat
               {
                  E = E + 1;
               }
               Until (in_buffer(E) || E == S);

               While (in_buffer(E) || E == S)
               {
                     D = D - 1;
                     E = E + 1;
               }
           }
        } /*End of ElseIf (S > E)*/

        F[D] = F[D] + 1; /*Update frequency for displacement D.*/
```

3. Normalize F[i] to get RD[i] for all values of i (0 <= i <= DT) using

$$RD[i] = \frac{F[i]}{\text{Sum}(F[j] \text{ for } 0 \le j \le DT)}$$

The algorithm starts with initialization of D to 0 and E to 1. Let S be the sequence number of an arrived packet.

If S has been received previously or delayed subject to displacement threshold condition (case a), it is discarded.

If S is the expected packet (case 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 held 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 in-sequence waiting packets 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 (case 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 that has been 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, similar to what is done in the S=E case (in case b).

Frequency value for new value of displacement is incremented as shown in the algorithm.

Once the algorithm deals with all the packets and the frequency F[D] is computed, for all D, the F[D] values are normalized to get the density with respect to D. This function is called Reorder Density function.

4. Examples

We consider a few different sequences to exemplify above algorithm.

a. Case of no packet loss:

Consider a sequence of 5 packets (1,4,2,5,3) with DT = 10.

Table 1 and 2 show the computation steps when RD algorithm is applied to above sequence.

Table 1: Reorder Histogram computation steps

E	1	2	2	3	3
S	1	4	2	5	3
D	0	1	1	2	0
F[D]	1	1	2	1	2

(E,S,D,F[D] as described in section 3)

The last row (F[D]) represents the current frequency of occurrence of displacement D. The final set of values for F[D] are shown in table 2.

When first packet with sequence number S=1 arrives, it is same as expected sequence number E=1, resulting in displacement D=0. Next, when packet S=4 arrives instead of expected packet E=2, the displacement D becomes 1. After receiving packet with sequence number 2, the displacement D is still 1, since the packet 3 that is expected now is not yet received. Packet 4 continues to occupy a buffer. Only

one buffer is needed and hence D = 1. On receiving packet with sequence number 5, the displacement D becomes 2. Finally, when we receive packet with sequence number 3, all the packets upto sequence

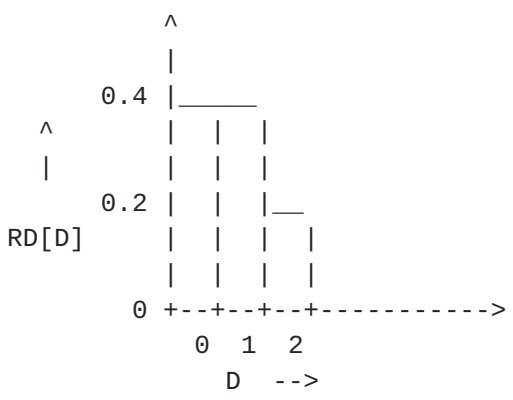
number 5 have been received. Thus the buffers can be released and hence the displacement D becomes 0. The reorder density function (RD) is derived by normalizing reorder histogram in Table 1 as follows:

Table 2: Reorder Density Function (RD)

D	0	1	2
F[D]	2	2	1
RD[D]	0.4	0.4	0.2

(D,F[D],RD[D] as described in section 3)

Graphical representation of above RD is as follows:



b. Case of packet loss:

Consider a sequence of 6 packets (1,2,4,5,6,7) with DT = 3.

Table 3 and 4 show the computation steps when RD algorithm is applied to above sequence.

Table 3: Reorder Histogram computation steps

E	1	2	3	3	3	3
S	1	2	4	5	6	7
D	0	0	1	2	3	0
F[D]	1	2	1	1	1	3

(E,S,D,F[D] as described in section 3)

When packet with sequence number 4 is received, the expected packet E is 3. So the displacement D increases by 1. When packets with sequence numbers 5 and 6 arrive, displacement D increases to 2 and

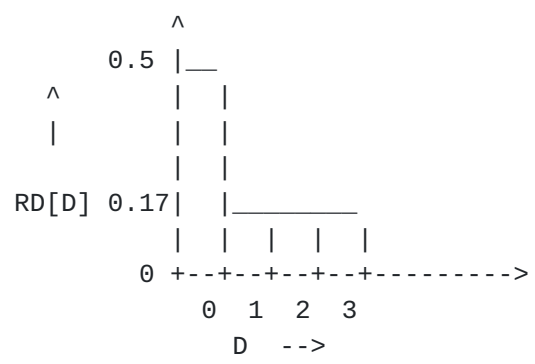
then to 3 respectively. Displacement is now equal to the displacement threshold DT=3. Therefore, when packet 7 is received, we no longer expect packet with sequence number 3 to arrive and assume that it is lost. We can now use all the waiting packets (4,5,6 and 7), reducing

the displacement to 0. The reorder density function (RD) is derived by normalizing reorder histogram in Table 3 as follows:

Table 4: Reorder Density Function (RD)				
D	0	1	2	3
F[D]	3	1	1	1
RD[D]	0.5	0.17	0.17	0.17

(D,F[D],RD[D] as described in section 3)

Graphical representation of above RD is as follows.



c. Case of Duplicate packets:

Consider a sequence of 6 packets (1,3,2,3,4) with DT = 5.

Table 5 and 6 show the computation steps when RD algorithm is applied to above sequence.

Table 5: Reorder Histogram computation steps					
E	1	2	2	4	4
S	1	3	2	3	4
D	0	1	0	0	0
F[D]	1	1	2	3	4

(E,S,D,F[D] as described in section 3)

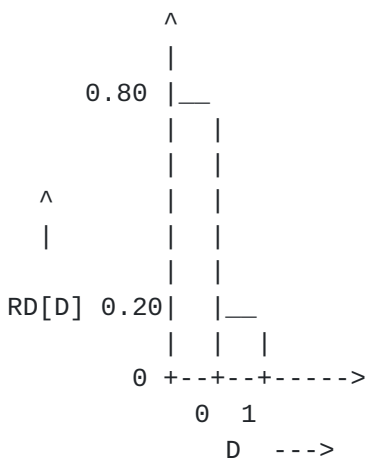
In the above sequence, duplicate packets are received by destination. RD algorithm ignores the duplicate arrival of the packets.

The reorder density function (RD) is derived by normalizing reorder histogram in Table 5 as follows:

Table 6: Reorder Density Function (RD)		
D	0	1
F[D]	4	1
RD[D]	0.80	0.20

(D,F[D],RD[D] as described in section 3)

Graphical Representation of RD is as follows:



5. 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. One may use following parameters derived from the reorder density that can act as metrics for packet reordering.

5.1 90th percentile

This parameter is the displacement value, such that 90 % of the arrived packets have displacement less than this value.

5.2 Median

Half of the arrived packets have displacements less than the median and the rest have displacement value greater than the median.

5.3 Mean and Standard Deviation

Mean and standard deviation of the displacements of arrived packets may be used as simple metrics.

6. Security Considerations

This document doesn't define any protocol. The metric definition per se is believed to have no security implications.

7. IANA Considerations

This document requires nothing from IANA.

8. References

S. Shalunov, "Definition of IP Packet Reordering Metric", Internet Draft, <draft-shalunov-reordering-definition-00.txt>, December 2001.

A. Morton, L. Ciavattone and G. Ramachandran, "Reordering Metric for IPPM using Non-Reversing Sequence", Internet Draft, <draft-morton-ippm-nonrev-reordering-00.txt>, February 2002.

D. Pullin, A. Corlett, S. Critchley and B. Mandeville, "Packet Reordering: The Minimal Longest Ascending Subsequence Metric", Internet Draft, <draft-critchley-mlas-reordering-00.txt>, February 2002.

T. Banka, A. A. Bare, A. P. Jayasumana, "Metrics for Degree of Reordering in Packet Sequences", Proc. 27th IEEE Conference on Local Computer Networks, Tampa, FL, Nov. 2002.

9. Authors' Addresses

Anura Jayasumana <anura@engr.colostate.edu>
Nischal M. Piratla <nischal@engr.colostate.edu>
Abhijit A. Bare <abhijit@cs.colostate.edu>
Tarun Banka <tarunb@cs.colostate.edu>

Computer Network Research Laboratory,
Department of Electrical and Computer Engineering,
Colorado State University,
Fort Collins, CO 80523.

Expiration Date: August 2003