Internet Engineering Task Force
INTERNET-DRAFT
draft-jayasumana-reorder-density-05.txt

Anura P. Jayasumana
Nischal M. Piratla
Abhijit A. Bare
Tarun Banka
Colorado State University
Rick Whitner
Jerry McCollom
Agilent Technologies

September 2005
Expires: March 2006

Reorder Density and Reorder Buffer-occupancy Density - Metrics for Packet Reordering Measurements

SStatus of this Memo

Copyright Notice

Abstract

Out of order arrival of packets is a common occurrence on the Internet, and it will be more widespread as link speeds increase. Percentage packet reordering as a metric of reordering is vague and unclear. A good reorder metric will capture the occurrence and characteristics of reordering comprehensively, and will have broader

utility than merely distinguishing among different reordered sequences.

Two metrics for packet reordering are presented, namely, Reorder Density (RD) and Reorder Buffer-occupancy Density (RBD). A threshold is used to clearly define when a packet is considered lost, to bound computational complexity at O(N), and to keep the memory requirement for evaluation independent of N, where N is the length of the packet sequence. RD is a comprehensive metric that captures the characteristics of reordering, while RBD evaluates the sequences from the point of view of recovery from reordering. These metrics are simple to compute yet comprehensive in their characterization of packet reordering. The measures are robust and orthogonal to packet loss and duplication.

Table of Contents

1. Introduction and Motivation

Packet reordering is a phenomena that occurs in Internet Protocol (IP) networks. Major causes of packet reordering include, but are not limited to, packet stripping at layers 2 and 3 [1,2], priority scheduling (e.g., diffserv), and route fluttering [3,4,5]. Reordering leads to degradation of the performance of many applications [1,6,7]. For example, perceived voice quality degrades if a Voice over IP (VoIP)application receives packets out-of-order. Increased link speeds, increased parallelism within routers and switches, QoS support, and load balancing among links all point to future networks with increased packet reordering. In order to understand and mitigate the effects of order, effective reordering metrics are required. Once the reordering in arriving packet stream is measured and quantified, it might be possible to predict the effects of reordering on applications that are sensitive to reordering, and perhaps even compensate for reordering. A metric for reordered packets might also help evaluate network protocols and implementations with respect to their impact on packet reordering.

The percentage of out-of-order packets is often used as a metric for characterizing reordering. However, this metric is vague and lacks in detail. Further, there is no uniform definition of the degree of reordering of an arrived packet [8,9]. For example, consider the two packet sequences (1, 3, 4, 2, 5) and (1, 4, 3, 2, 5). It is possible to interpret the reordering of packets in these sequences differently.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 the first sequence, while packets 4 and 2 are out-of-order in the second sequence.

In essence, the percentage of out-of-order packets as a metric of reordering is subject to interpretation and cannot capture the reordering unambiguously and hence, accurately.

Other metrics attempt to overcome this ambiguity by defining only the late packets or only the early packets as being reordered. However, measuring reordering based on only late or early packets is not always effective. Consider, for example, a sequence of packets with the only anomaly being that packet 20 is delivered immediately after packet 1, i.e., the sequence (1, 20, 2, 3,..,19, 21, 22, ...). A metric based only on lateness will indicate a high degree of reordering, even though in this example it is a single packet arriving ahead of others. Similarly, a metric based only on earliness

does not accurately capture reordering caused by a late arriving packet. A complete reorder metric must account for both earliness and lateness, and must be able to differentiate between the two. Inability to capture both earliness and lateness precludes a metric from being used for estimating end-to-end reordering based on reordering in constituent subnets.

There are other questions regarding what constitutes a good reordering metric. Consider again the packet sequence (1, 3, 4, 2, 5). From an operational view, if buffers are available to store packets 3 and 4 while waiting for packet 2, an application can recover from the reordering and the reordering is effectively insignificant. However, there might be cases where an application behaves such that arrival of packet 2 out of order renders the packet useless. In this case reordering can be very significant. While one can argue that a good packet reordering measurement scheme should capture application-specific effects, a counter argument can be made that packet reordering should be measured strictly with respect to the order of delivery and should be independent of the application.

The desirable attributes of a packet reorder metric include:

1) Simplicity: The metric should be simple yet contain sufficient information to be useful.
2) Orthogonality: The metric, to the extent possible, should be independent of or orthogonal to other phenomena that affect the packet streams, e.g., packet loss and duplication.
3) Usefulness: Rather than being a mere representation of the amount of reordering in a packet stream, a reorder metric should be useful to the application and to resource management schemes. For example,it should allow one to determine the size of buffer that is required to recover from the effects of reordering.
4) Differentiability: The metric should provide insight into the nature and severity of reordering and perhaps even into possible causes.
5) Computational complexity: The metric should be able to be computed in real-time. When evaluating reordering in an arbitrarily long sequence, it should be possible to keep a running measurement without having to wait until all packets have arrived. The memory requirement, i.e., to retain the necessary state information, should not grow with the length of the sequence (N), and the computation time should be O(N).
6) Robustness: The metric should be self-correcting against events such as bursty losses and sequence number wraparound.
7) Proportionality: The metric should have a sense of proportionality, i.e., the metric should not change significantly due to the peculiar behavior of a very small number of packets. For example, in a Transmission Control Protocol (TCP) sequence

number rollover scenario, a single rogue packet with a high

sequence number from the previous measurement cycle must not significantly skew the metric when it appears in the initial part of the next measurement cycle.

8) Broad Applicability: A good metric should have applicability beyond just characterizing the nature of reordering in a given sequence of packets. For example, a good metric might allow one to combine the reorder characteristics of individual adjacent networks to predict the reorder behavior of the concatenation of these networks.

In this memo, we define two density functions, Reorder Density (RD) and Reorder Buffer-occupancy Density (RBD), that capture the nature of reordering in a packet stream. These two metrics can be used individually or collectively to characterize the reordering in a packet stream.

RD is the normalized distribution of displacements of packets from their original positions. Lost and duplicate packets are accounted for when evaluating these displacements. The nature of reordering introduced by a network with stationary statistical characteristics can be captured using this metric in the form of a reorder response [9,10]. For reordering introduced by such a system, or for a statistically significant sequence of packets, RD is the probability density of the packet displacement.RD measured on individual subnets can be combined for to predict the end-to-end reorder characteristics of the network formed by the cascade of subnets under a fairly broad set of conditions [10].

RBD is the normalized histogram of the occupancy of a hypothetical buffer that would allow the recovery from out-of-order delivery of packets. If an arriving packet is early, it is added to a hypothetical buffer until it can be released in order. The occupancy of this buffer after each arrival is used as the measure of reordering. In situations where the late arrival of a packet might be regarded as useless, e.g., a real-time application, a threshold on the hypothetical buffer size is defined, as explained in section 2.10. In [8], this metric was called RD and buffer occupancy was known as displacement.

RD and RBD are simple metrics that are useful to evaluate and improve application performance. These metrics are robust against peculiarities as highlighted previously, and have a computational complexity of O(N), where the received sequence size is N. RD is orthogonal to loss and duplication, whereas RBD is orthogonal to duplication. Finally, RD of a network formed by the cascade of two subnets is equivalent to the convolution of the RDs of the individual subnets.

2. Definitions

The following terms are used to describe RD, RBD, and the measurement algorithm. Wraparound of sequence numbers is not explicitly addressed in this document, but with the use of modulo-N arithmetic, all claims made here remain valid in the presence of wraparound.

2.1 Receive_index (RI)

Consider a sequence of packets (1, 2, ..., N) transmitted over a network. A receive_index RI, e.g., (1, 2, ...), is a value assigned to a packet as it arrives at its destination. A receive_index is not assigned to duplicate packets, and the receive_index value skips the value corresponding to a lost packet. (The detection of loss and duplication for this purpose is described in section 5.) In the absence of reordering the sequence number of the packet and the receive_index are same for each packet.

RI is used to compute earliness and lateness of an arriving packet. Below are two examples of received sequences with receive_index values for a sequence of 5 packets (1, 2, 3, 4, 5) arriving out of order:

Example 1:

Arrived sequence:	2	1	4	5	3
Receive_index:	1	2	3	4	5

Example 2:

Arrived sequence:	1	4	3	5	3
Receive_index:	1	3	4	5	-

In example 1, there is no loss or duplication. In example 2, the packet with sequence number 2 is lost, thus 2 is not assigned as an RI; packet 3 is duplicated, thus the second copy is not assigned an RI.

2.2 Out-of-Order Packet

When the sequence number of a packet is not equal to the RI assigned to it, it is considered an out-of-order packet. Duplicates for which an RI is not defined are ignored.

2.3 Early-packet and Late-packet

An arriving packet is early if its sequence number is greater than its RI. An arriving packet is late if its sequence number is less than its RI. Let the receive_index of arriving packet i be RI[i]. If i > RI[i] then the packet is early. If i < RI[i] then the packet

is late.

2.4 Displacement (D)

Displacement of a packet is defined as the difference between RI and the sequence number of the packet, i.e., the displacement of packet i is RI[i] - i. Thus, a negative displacement refers to the earliness of a packet and a positive displacement to the lateness. In example 3 below, an arrived sequence with displacements of each packet is illustrated.

Example 3:

Arrived sequence:	1	4	3	5	3	8	7	6
Receive_index:	1	3	4	5	-	6	7	8
Displacement:	0	-1	1	0	-	-2	0	2

2.5 Displacement Threshold (DT)

The displacement threshold is a threshold on the displacement of a packet that allows the metric to classify a packet as lost or duplicate. Determining when to classify a packet as lost is difficult because there is no point in time at which a packet can theoretically be classified as lost; the packet might still arrive after some arbitrarily long delay. However, from a practical point of view, a packet may be classified as lost if it has not arrived within a certain administratively defined displacement threshold, DT. Similarly, to identify a duplicate packet, it is theoretically necessary to keep track of all arrived (or missing) packets. Again, however, from a practical point of view, missing packets within a certain window of sequence numbers suffice. Thus, DT is used as a practical means for declaring a packet as lost or duplicated. DT makes the metric more robust, keeps the computational complexity for long sequences within O(N), and keeps storage requirements independent of N.

To be effective, the choice of DT is critical. If DT is selected too small, reordered packets might be classified as lost. A large DT will increase both the size of memory required to keep track of sequence numbers and the length of computation time to required to evaluate the metric. Indeed, it is possible to use two different thresholds for the two cases. The selection of DT is further discussed in section 4.

2.6 Lateness/Earliness Frequency (FLE)

Lateness/Earliness Frequency FLE[k] is the number of arrived packets having a displacement of k, where k takes values from -DT to DT.

2.7 Reorder Density (RD)

RD is defined as the distribution of all Lateness/Earliness Frequencies FLE[k] normalized with respect to N' total number of non-duplicate packets received, where N' is the length of the received sequence, ignoring lost and duplicate packets. N' is also the sum(FLE[k]) for all k such that k belongs to [-DT, DT].

2.8 Expected Packet (E)

A packet with sequence number E is expected if E is the largest number such that all packets with sequence number less than E have already arrived or have been determined to be lost.

2.9 Buffer Occupancy (B)

An arrived packet with a sequence number greater than that of an expected packet is considered to be stored in a hypothetical buffer sufficiently long to permit recovery from reordering. At any packet arrival, the buffer occupancy is equal to the number of out-of-order packets in the buffer, including the arrived packet (assuming one buffer location for each packet). For example, for the sequence of packets (1, 2, 4, 5, 3) with expected order (1, 2, 3, 4, 5), when packet 4 arrives the buffer occupancy is 1 because packet 4 arrived early. Similarly, the buffer occupancy becomes 2 when packet 5 arrives. When packet 3 arrives, recovery from reordering occurs and the buffer occupancy reduces to zero.

2.10 Buffer Occupancy Threshold (BT)

Buffer occupancy threshold is a threshold on the maximum size of the hypothetical buffer that is used for recovery from reordering. As in the case of DT, BT is used for loss and duplication classification for Reorder Buffer-occupancy Density (RBD) computation (see section 2.12). BT provides robustness, and limits the computation complexity of RBD.

2.11 Buffer Occupancy Frequency (FB)

At the arrival of each packet the buffer occupancy may take any value k ranging from 0 to BT. The buffer occupancy frequency FB[k] is the number of times the occupancy takes the value of k.

2.12 Reorder Buffer-Occupancy Density (RBD)

Reorder buffer-occupancy density is the buffer occupancy frequencies normalized by the total number of non-duplicate packets, i.e., RBD[k] = FB[k]/N' where N' is the length of the received sequence, ignoring excessively delayed (deemed lost) and duplicate packets. N'

is also the sum(FB[k]) for all k such that k belongs to [0, BT].

3. Representation of Packet Reordering and Reorder Density

Consider a sequence of packets (1, 2, ..., N). Let the RI assigned to packet m be the sequence number m plus some non-negative offset dm, i.e., (m + dm). A reorder event of packet m is represented by r(m, dm). When dm is not equal to zero, a reorder event is said to have occurred. A packet is late if dm > 0 and early if dm < 0. Thus, packet reordering of a sequence of packets is completely represented by the union of reorder events, R, referred to as the reorder set:

R = {r(m,dm)| dm not equal to 0 for all m}

If there is no reordering in a packet sequence then R is the null set.

Examples 4 and 5 illustrate the reorder set:

Example 4. No losses or duplicates

Arrived Sequence	1	2	3	5	4	6
Receive_index	1	2	3	4	5	6
Displacement	0	0	0	-1	1	0

R = {(4,1), (5,-1)}

Example 5. Packet 4 is lost and 2 is duplicated

Arrived Sequence	1	2	5	3	6	2
Receive_index	1	2	3	5	6	-
Displacement	0	0	-2	2	0	-

R = {(3, 2), (5, -2)}

RD is defined as the discrete density of the frequency of packets with respect to their displacements, i.e., the lateness and earliness from the original position. Let S[k] denote the set of reorder events in R with displacement equal to k, i.e.,

S[k]= {r(m, dm)| dm = k}

Let |S[k]| be the cardinality of set S[k]. Thus, RD[k] is defined as |S[k]| normalized with respect to the total number of received packets (N'). Note that N' does not include duplicates or lost packets.

RD[k] = |S[k]| / N' for k not equal to zero.

RD[0] corresponds to the packets for which RI is the same as the sequence number:

$$RD[0] = 1 - sum(|S[k]| / N')$$

As defined previously, FLE[k] is the measure that keeps track of |S[k]|.

4. Selection of DT

Although assigning a threshold for determining lost and duplicate packets might appear to introduce error into the reorder metrics, in practice this need not be the case. Applications, protocols, and the network itself operate within finite resource constraints which introduce practical limits beyond which the choice of certain values become irrelevant. In the case of DT, it is common to find a value above which DT does not have an impact on the reorder metrics. For example, in case of a VoIP application with a bit-rate of 128kbps and packet size of 200 bytes, a practical DT value can be determined as follows. Assume that the application can wait a maximum of 50 ms for an expected packet and that the packets arrive at constant rate. Within 50 ms, the application can receive (128*1000*0.05)/(200*8), i.e., 4 packets. Since packets arriving after this duration are effectively lost, the DT value could be set at 4. If the operational nature of an application is such that a DT can be defined, then using DT in the computation of reorder metrics will not invalidate nor limit the effectiveness of the metrics, i.e., increasing DT does not provide any benefit. In the case of TCP, the transmit and receive window sizes impose a natural limit on the useful value of DT.

If there are no operational constraints imposed by factors as described above, or if one is purely interested in a more complete picture of reordering, then DT can be made as large as required. If DT is equal to the length of the packet sequence (worst case scenario), a complete picture of reordering is seen. This requires that either the length of the packet sequence is known beforehand, or that DT be allowed to grow without bound.

5. Detection of Lost and Duplicate Packets

The RD and RBD algorithms compare the sequence number of arriving packets against the expected sequence number E and against sequence numbers stored in a buffer. Only sequence numbers for early arrivals,i.e., those with sequence numbers greater than E, are stored. For both RD and RBD, this buffer size is limited by the thresholds DT and BT, respectively.

5.1 Detection of Duplicate Packets

Non-duplicate arriving packets do not have a copy in the buffer and do not have a sequence number less (earlier) than E.

5.2 Detection of Lost Packets

In RD, a packet is not considered lost until it is late beyond DT. The question arises as to how to assign an RI to packets with later packet numbers. This can be handled in one of two ways:

a) Go-back Method: RD is computed as packets arrive. When a packet is deemed lost, RI values are corrected and displacements recomputed. The Go-back Method is only invoked when a packet is lost.

b) Stay-back Method: RD evaluation lags the arriving packets so that the correct RI and E values can be assigned to each packet as it arrives. Here, RI is assigned to a packet only once, and the value assigned is guaranteed to be correct. In the worst case, the computations are delayed by DT packets. The lag associated with the Stay-back Method is incurred only when a packet is missing.
In RBD, a packet is considered lost if the buffer is filled to its threshold BT. At this point the expected is incrementing and buffer contents may be emptied, if neccessary.

6. Algorithms to compute RD and RBD

The algorithms to compute RD and RBD are given below. For simplicity, the sequence numbers start from 1 and continue in increments of 1. Only the Stay-back Method of loss detection is presented here, hence the RD values lag by a maximum of DT. Both Stay-back and Go-back methods are described in [9]. Perl scripts for these algorithms are posted in [11].

6.1 RD Algorithm

Variables used:

```
-------------------------------------------------------------------
 RI: receive_index.
 S: Arrival under consideration for lateness/earliness computation.
 D: Lateness or earliness of the packet being processed.
 FLE[ -DT..DT]: Frequency of lateness and earliness.
 window[1..DT+1]: List of incoming sequence numbers.
 buffer[1..DT]: Array to hold sequence numbers of early arrivals.
 window[] and buffer[] are empty at the beginning.
===================================================================
```

```
   Step 1. Initialize:

        Store first unique DT+1 sequence numbers in arriving order into
        window;
        RI = 1;

   Step 2. Repeat:

        If (window or buffer contains sequence number RI)
        {
           Copy first sequence number in window to S;
           Delete first sequence number from window;
           D = RI - S; # compute displacement

           If (absolute(D) <= DT) # Apply threshold
           {
              FLE[D]++; # Update frequency

              If (buffer contains sequence number RI)
                 Delete RI from buffer;

              If (D < 0) # Early Arrival
                 add S to empty slot in buffer;
              RI++; # Update RI value
           }

           Else # Displacement beyond threshold.
           {
              Discard S;
           }
           # Get next incoming non-duplicate sequence number, if any.
           newS = get_next_arrival(); # subroutine called*
           if (newS != null)
           {
                add newS to window;
           }
           if (window is empty) go to step 3;
        }
        Else # RI not found. Get next RI value.
        {
           # Next RI is the minimum among window and buffer contents.
           m = minimum (minimum (window), minimum (buffer));
           If (RI < m)
              RI = m;
           Else
              RI++;
        }

   Step 3. Normalize FLE to get RD;
```

```
* Get a new sequence number from packet stream, if any
subroutine get_next_arrival()
{
     do   # get non-duplicate next arrival
     {
           newS = new sequence from arriving stream;
           if (newS == null) # End of packet stream
              return null;
     } while (newS < RI or newS in buffer or newS in window);

     return newS;
}
```

6.2 RBD Algorithm

```
Variables used:
-------------------------------------------------------------------
# E : Next expected sequence number.
# S : Sequence number of the packet just arrived.
# B : Current buffer occupancy.
# BT: Buffer Occupancy threshold.
# FB[i]: Frequency of buffer occupancy i  (0 <= i <= BT).
# in_buffer(N) : True if the packet with sequence number N is
  already stored in the buffer.
===================================================================

1.  Initialize E = 1, B = 0 and FB[i] = 0 for all values of i.
2.  Do the following for each arrived packet.
       If (in_buffer(S) || S < E) /*Do nothing*/;
       /* Case a: S is a duplicate or excessively delayed packet.
       Discard the packet.*/
       Else
       {

          If (S == E)
          /* Case b: Expected packet has arrived.*/
          {
             E = E + 1;
             While (in_buffer(E))
             {
                B = B - 1; /* Free buffer occupied by E.*/
                E = E + 1; /* Expect next packet.*/
             }
             FB[B] = FB[B] + 1; /*Update frequency for buffer
             occupancy B.*/
          } /* End of ElseIf (S == E)*/
```

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

                   While (in_buffer(E) || E == S)
                   {
                      if (E != S) B = B - 1;
                      E = E + 1;
                   }
                 }
                 FB[B] = FB[B] + 1; /*Update frequency for buffer
                 occupancy B.*/
             } /* End of ElseIf (S > E)*/

          }
```

3. Normalize FB[i] to obatin RBD[i], for all values of i using

$$RBD[i] = \frac{FB[i]}{\text{Sum}(FB[j] \text{ for } 0 <= j <= BT)}$$

7. Examples

a. Scenario with no packet loss

Consider the sequence of packets (1, 4, 2, 5, 3, 6, 7, 8) with DT = BT = 4.

Tables 1 and 2 show the computational steps when the RD algorithm is applied to the above sequence.

Table 1: Late/Early-packet Frequency computation steps

S	1	4	2	5	3	6	7	8
RI	1	2	3	4	5	6	7	8
D	0	-2	1	-1	2	0	0	0
FLE[D]	1	1	1	1	1	2	3	4

(S, RI,D and FLE[D] as described in section 6.1)

The last row (FLE[D]) represents the current frequency of occurrence of the displacement D, e.g., column 3 indicates FLE[1] = 1 while column 4 indicates FLE[-1] = 1. The final set of values for RD are shown in Table 2.

Table 2: Reorder Density (RD)

D	-2	-1	0	1	2
FLE[D]	1	1	4	1	1
RD[D]	0.125	0.125	0.5	0.125	0.125

(D,FLE[D] and RD[D] as described in section 6.1)

Tables 3 and 4 illustrate the computational steps for RBD for the same example.

Table 3: Buffer occupancy frequencies (FB) computation steps

S	1	4	2	5	3	6	7	8
E	1	2	2	3	3	6	7	8
B	0	1	1	2	0	0	0	0
FB[B]	1	1	2	1	2	3	4	5

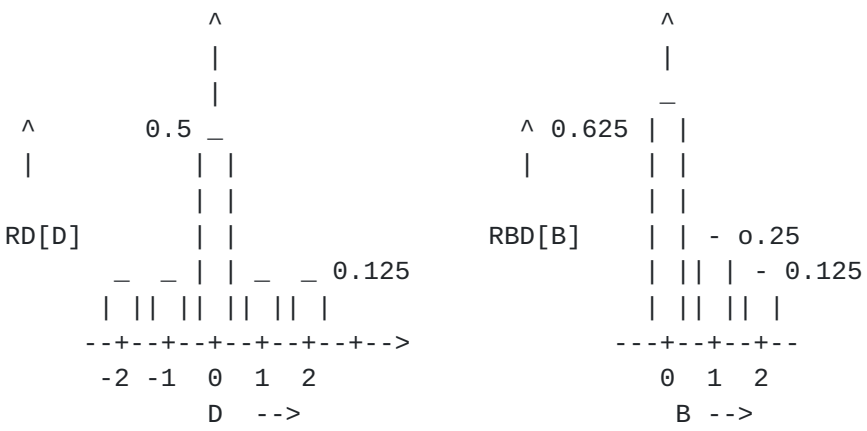
(E,S,B and FB[B] as described in section 6.2)

Table 4: Reorder Buffer-occupancy Density

B	0	1	2
FB[B]	5	2	1
RBD[B]	0.625	0.25	0.125

(B,FB[B] and RBD[B] as discussed in section 6.2)

Graphical representations of the densities are as follows:



b. Scenario with packet loss

Consider a sequence of 6 packets (1, 2, 4, 5, 6, 7) with DT = BT = 3. Table 5 shows the computational steps when the RD algorithm is applied to the above sequence to obtain FLE[D].

Table 5: Late/Early-packet Frequency computation steps

S	1	2	4	5	6	7
RI	1	2	4	5	6	7
D	0	0	0	0	0	0
FLE[D]	1	2	3	4	5	6

(S,RI,D and FLE[D] as described in section 6.1)

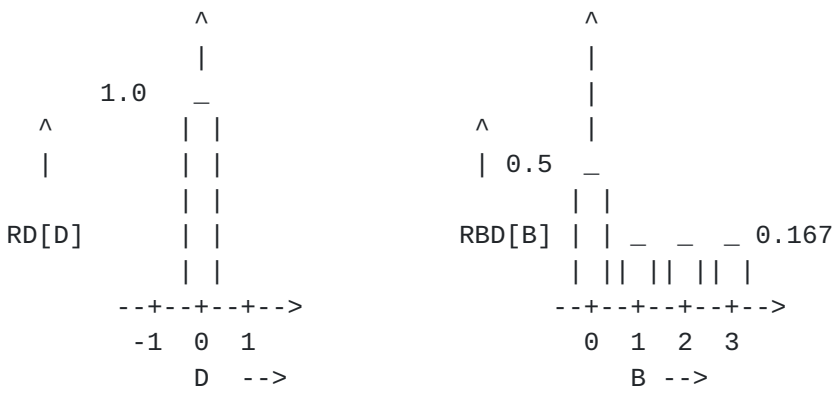
Table 6 illustrates the FB[B] for the above arrival sequence.

Table 6: Buffer occupancy computation steps

S	1	2	4	5	6	7
E	1	2	3	3	3	7
B	0	0	1	2	3	0
FB[B]	1	2	1	1	1	3

(E,S,B and FB[B] as described in section 6.2)

Graphical representations of RD and RBD for the above sequence are as follows.



c. Scenario with duplicate packets

Consider a sequence of 6 packets (1, 3, 2, 3, 4, 5) with DT = 2. Tables 7 shows the computational steps when the RD algorithm is applied to the above sequence to obtain FLE[D].

Table 7: Late/Early-packet Frequency computation steps

S	1	3	2	3	4	5
RI	1	2	3	-	4	5
D	0	-1	1	-	0	0
FLE[D]	1	1	1	-	2	3

(S, RI,D and FLE[D] as described in section 6.1)

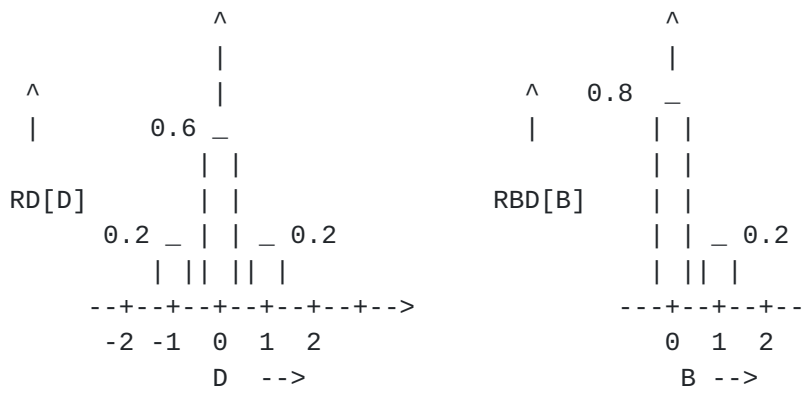
Table 8 illustrates the FB[B] for the above arrival sequence.

Table 8: Buffer Occupancy Frequency computation steps

S	1	3	2	3	4	5
E	1	2	2	-	4	5
B	0	1	0	-	0	0
FB[B]	1	1	2	-	3	4

(E,S,B and FB[B] as described in section 6.2)

Graphical representations of RD, RBD and RBDLO for the above sequence are as follows:



8. Comparison with Other Metrics

RD and RBD are compared to other metrics that are being proposed [12] in [15]. This section is for review purposes only and will be removed from the final draft.

9. Security Considerations

This document does not define any protocol. The metric definition per se is believed to have no security implications.

10.IANA Considerations

This document requires nothing from the IANA.

11.References

[1] J. C. R. Bennett, C. Partridge and N. Shectman, "Packet Reordering is Not Pathological Network Behavior," Trans. on Networking IEEE/ACM, Dec. 1999, pp.789-798.

[2] S. Jaiswal, G. Iannaccone, C. Diot, J. Kurose and D. Towsley, "Measurement and Classification of Out-of-sequence Packets in Tier-1 IP Backbone," Proc. IEEE INFOCOM, Mar. 2003, pp. 1199-1209.

[3] V.Paxson, "Measurements and Analysis of End-to-End Internet Dynamics," Ph.D. dissertation, U.C. Berkeley, 1997, ftp://ftp.ee.lbl.gov/papers/vp-thesis/dis.ps.gz.

[4] S. Bohacek, J. Hespanha, J. Lee, C. Lim and K.Obraczka, "TCP-PR: TCP for Persistent Packet Reordering," In Proc. of the IEEE 23rd ICDCS, May 2003, pp.222-231.

[5] G. Iannaccone, S. Jaiswal and C. Diot, "Packet Reordering Inside the Sprint Backbone," Tech.Report, TR01-ATL-062917, Sprint ATL, Jun. 2001.

[6] E. Blanton and M. Allman, "On Making TCP More Robust to Packet Reordering," ACM Computer Comm. Review, 32(1), Jan. 2002, pp.20-30.

[7] M. Laor and L. Gendel, "The Effect of Packet Reordering in a Backbone Link on Application Throughput," IEEE Network, Sep./Oct. 2002, pp.28-36.

[8] 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] N. M. Piratla, "A Theoretical Foundation, Metrics and Modeling of Packet Reordering and Methodology of Delay Modleing using Inter-packet Gaps," Ph.D. Dissertation, Department of Electrical and Computer Engineering, Colorado State University, Fort Collins, CO, Fall 2005.

[10] N. M. Piratla, A. P. Jayasumana and A. A. Bare, "RD: A Formal, Comprehensive Metric for Packet Reordering," Proceedings, International IFIP-TC6 Networking Conference (Networking 2005), Waterloo, Canada, May 2-6, 2005, LNCS 3462, pp: 78-89.

[11] Perl Scripts for RLED and RBD, http://www.cnrl.colostate.edu/Reorder_Density.html, Last modified on Jul. 18, 2004.

[12] A. Morton, L. Ciavattone, G. Ramachandran, S.Shalunov and J.Perser, "Packet Reordering Metric for IPPM", Internet Draft, <draft-ietf-ippm-reordering-08.txt>, December 2004.

[13] M. Zhang, B. Karp, S. Floyd and L. Peterson, "RR-TCP: A Reordering-Robust TCP with DSACK," Proc. The Eleventh IEEE International Conference on Networking Protocols (ICNP 2003), Atlanta, GA, Nov. 2003, pp. 95-106.

[14] A. A. Bare, "Measurement and Analysis of Packet Reordering Using Reorder Density," Masters Thesis, Department of Computer Science, Colorado State University, Fort Collins, Colorado, Fall 2004.

[15] N. M. Piratla, A. P. Jayasumana and A. A. Bare, "A Comparative Analysis of Packet Reordering Metrics," To Appear in Proc. COMSWARE, New Delhi, India, Jan. 2006.

[16] N. M. Piratla, A. P. Jayasumana and T. Banka, "On Reorder Density and its Application to Characterization of Packet Reordering," To appear in Proc. 30th IEEE Local Computer Networks Conference (LCN 2005), Sydney, Australia, Nov. 2005.

12. Authors' Addresses

Anura P. Jayasumana <Anura.Jayasumana@colostate.edu>
Nischal M. Piratla <Nischal.Piratla@colostate.edu> *
Abhijit A. Bare <abhijit_bare@agilent.com>
Tarun Banka <Tarun.Banka@colostate.edu>
Computer Networking Research Laboratory,
Department of Electrical and Computer Engineering,
1373 Colorado State University,
Fort Collins, CO 80523

* (Effective Oct. 17, 2005)
Deutsche Telekom Laboratories
Ernst-Reuter-Platz 7,
D-10587 Berlin, Germany

Rick Whitner <rick_whitner@agilent.com>
Jerry McCollom <jerry_mccollom@agilent.com>
Agilent Technologies, 4380 Ziegler Rd.,
Fort Collins, CO 80525

Expiration Date: March 2006

Full Copyright Statement

Intellectual Property

The IETF takes no position regarding the validity or scope of any Intellectual Property Rights or other rights that might be claimed to pertain to the implementation or use of the technology described in this document or the extent to which any license under such rights might or might not be available; nor does it represent that it has made any independent effort to identify any such rights. Information on the procedures with respect to rights in RFC documents can be found in BCP 78 and BCP 79.

Copies of IPR disclosures made to the IETF Secretariat and any assurances of licenses to be made available, or the result of an attempt made to obtain a general license or permission for the use of such proprietary rights by implementers or users of this specification can be obtained from the IETF on-line IPR repository at http://www.ietf.org/ipr.

The IETF invites any interested party to bring to its attention any copyrights, patents or patent applications, or other proprietary rights that may cover technology that may be required to implement this standard. Please address the information to the IETF at ietf-ipr@ietf.org.