Network Working Group   A. Jayasumana
Request for Comments: 5236   Colorado State University
Category: Informational   N. Piratla
                                                   Deutsche Telekom Labs
                                                              T. Banka
                                               Colorado State University
                                                                A. Bare
                                                             R. Whitner
                                               Agilent Technologies, Inc.
                                                              June 2008

                      Improved Packet Reordering Metrics

Status of This Memo

IESG Note

Abstract

   This document presents two improved metrics for packet reordering,
   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 phenomenon that occurs in Internet Protocol
   (IP) networks. Major causes of packet reordering include, but are
   not limited to, packet striping at layers 2 and 3 [Ben99] [Jai03],
   priority scheduling (e.g., Diffserv), and route fluttering [Pax97]
   [Boh03]. Reordering leads to degradation of the performance of many
   applications [Ben99] [Bla02] [Lao02]. Increased link speeds [Bar04],
   increased parallelism within routers and switches, Quality-of-Service
   (QoS) support, and load balancing among links all point to increased
   packet reordering in future networks.

   Effective metrics for reordering are required to measure and quantify
   reordering. A good metric or a set of metrics capturing the nature
   of reordering can be expected to provide insight into the reordering
   phenomenon in networks. It may be possible to use such metrics to
   predict the effects of reordering on applications that are sensitive
   to packet reordering, and perhaps even to compensate for reordering.
   A metric for reordered packets may 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 lacking
   in detail. Further, there is no uniform definition for the degree of
   reordering of an arrived packet [Ban02] [Pi05a]. 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 only on late or only on early packets is
   not always effective. Consider, for example, the sequence (1, 20, 2,

3, ..., 19, 21, 22, ...); the only anomaly is that packet 20 is
delivered immediately after packet 1. 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 it
must be able to differentiate between the two. The inability to
capture both the earliness and the lateness precludes a metric from
being useful for estimating end-to-end reordering based on reordering
in constituent subnets.

The sensitivity to packet reordering can vary significantly from one
application to the other. Consider again the packet sequence (1, 3,
4, 2, 5). If buffers are available to store packets 3 and 4 while
waiting for packet 2, an application can recover from reordering.
However, with certain real-time applications, the out-of-order
arrival of packet 2 may render it useless. While one can argue that
a good packet reordering measurement scheme should capture
application-specific effects, a counter argument can also be made
that packet reordering should be measured strictly with respect to
the order of delivery, independent of the application.

Many different packet reordering metrics have been suggested. For
example, the standards-track document RFC 4737 [RFC4737] defines 11
metrics for packet reordering, including lateness-based percentage
metrics, reordering extent metrics, and N-reordering.

Section 2 of this document discusses the desirable attributes of any
packet reordering metric. Section 3 introduces two additional packet
reorder metrics: Reorder Density (RD) and Reorder Buffer-occupancy
Density (RBD), which we claim are superior to the others [Pi07]. In
particular, RD possesses all the desirable attributes, while other
metrics fall significantly short in several of these attributes. RBD
is unique in measuring reordering in terms of the system resources
needed for recovery from packet reordering. Both RD and RBD have a
computation complexity O(N), where N is the length of the packet
sequence, and they can therefore be used for real-time online
monitoring.

2. Attributes of Packet Reordering Metrics

The first and foremost requirement of a packet reordering metric is
its ability to capture the amount and extent of reordering in a
sequence of packets. The fact that a measure varies with reordering
of packets in a stream does not make it a good metric. In [Ben99],
the authors have identified desirable features of a reordering
metric. This list encloses the foremost requirements stated above:

   simplicity, low sensitivity to packet loss, ability to combine
   reorder measures from two networks, minimal value for in-order data,
   and independence of data size. These features are explained below in
   detail, along with additional desired features. Note, the ability to
   combine reorder measures from two networks is added to broaden
   applicability, and data size independence is discussed under
   evaluation complexity. However, data size independence could also
   refer to the final measure, as in percentage reordering or even a
   normalized representation.

   a) Simplicity

      An ideal metric is one that is simple to understand and evaluate,
      and yet informative, i.e., able to provide a complete picture of
      reordering. Percentage of packets reordered is the simplest
      singleton metric; but the ambiguity in its definition, as
      discussed earlier, and its failure to carry the extent of
      reordering make it less informative. On the other hand, keeping
      track of the displacements of each and every packet without
      compressing the data will contain all the information about
      reordering, but it is not simple to evaluate or use.

      A simpler metric may be preferred in some cases even though it
      does not capture reordering completely, while other cases may
      demand a more complex, yet complete metric.

      In striving to strike a balance, the lateness-based metrics
      consider only the late packets as reordered, and earliness-based
      metrics only the early packets as reordered. However, a metric
      based only on earliness or only on lateness captures only a part
      of the information associated with reordering. In contrast, a
      metric capturing both early and late arrivals provides a complete
      picture of reordering in a sequence.

   b) Low Sensitivity to Packet Loss and Duplication

      A reorder metric should treat only an out-of-order packet as
      reordered, i.e., if a packet is lost during transit, then this
      should not result in its following packets, which arrive in order,
      being classified as out of order. Consider the sequence (1, 3, 4,
      5, 6). If packet 2 has been lost, the sequence should not be
      considered to contain any out-of-order packets. Similarly, if
      multiple copies of a packet (duplicates) are delivered, this must

not result in a packet being classified as out of order, as long
as one copy arrives in the proper position. For example, sequence
(1, 2, 3, 2, 4, 5) has no reordering. The lost and duplicate
packet counts may be tracked using metrics specifically intended
to measure those, e.g., percentage of lost packets, and percentage
of duplicate packets.

c) Low Evaluation Complexity

Memory and time complexities associated with evaluating a metric
play a vital role in implementation and real-time measurements.
Spatial/memory complexity corresponds to the amount of buffers
required for the overall measurement process, whereas
time/computation complexity refers to the number of computation
steps involved in computing the amount of reordering in a
sequence. On-the-fly evaluation of the metric for large streams
of packets requires the computational complexity to be O(N), where
N denotes the number of received packets, used for the reordering
measure. This allows the metric to be updated in constant-time as
each packet arrives. In the absence of a threshold defining
losses or the number of sequence numbers to buffer for detection
of duplicates, the worst-case complexity of loss and duplication
detection will increase with N. The rate of increase will depend,
among other things, on the value of N and the implementation of
the duplicate detection scheme.

d) Robustness

Reorder measurements should be robust against different network
phenomena and peculiarities in measurement or sequences such as a
very late arrival of a duplicate packet, or even a rogue packet
due to an error or sequence number wraparound. The impact due to
an event associated with a single or a small number of packets
should have a sense of proportionality on the reorder measure.
Consider, for example, the arrival sequence: (1, 5430, 2, 3, 4, 5,
...) where packet 5430 appears to be very early; it may be due to
either sequence rollover in test streams or some unknown reason.

e) Broad Applicability

A framework for IP performance metrics [RFC2330] states: "The
metrics must aid users and providers in understanding the
performance they experience or provide".

Rather than being a mere value or a set of values that changes
with the reordering of packets in a stream, a reorder metric
should be useful for a variety of purposes. An application or a
transport protocol implementation, for example, may be able to use

the reordering information to allocate resources to recover from
reordering. A metric may be useful for TCP flow control, buffer
resource allocation for recovery from reordering and/or network
diagnosis.

The ability to combine the reorder metrics of constituent subnets
to measure the end-to-end reordering would be an extremely useful
property. In the absence of this property, no amount of
individual network measurements, short of measuring the reordering
for the pair of endpoints of interest, would be useful in
predicting the end-to-end reordering.

The ability to provide different types of information based on
monitoring or diagnostic needs also broadens the applicability of
a metric. Examples of applicable information for reordering may
include parameters such as the percentage of reordered packets
that resulted in fast retransmissions in TCP, or the percentage of
utilization of the reorder recovery buffer.

3. Reorder Density and Reorder Buffer-Occupancy Density

In this memo, we define two discrete 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. Also presented are algorithms for real-time
evaluation of these metrics for an incoming packet stream.

RD is defined as the distribution of displacements of packets from
their original positions, normalized with respect to the number of
packets. An early packet corresponds to a negative displacement and
a late packet to a positive displacement. A threshold on
displacement is used to keep the computation within bounds. The
choice of threshold value depends on the measurement uses and
constraints, such as whether duplicate packets are accounted for when
evaluating these displacements (discussed in Section 5).

The ability of RD to capture the nature and properties of reordering
in a comprehensive manner has been demonstrated in [Pi05a], [Pi05b],
[Pi05c], and [Pi07]. The RD observed at the output port of a subnet
when the input is an in-order packet stream can be viewed as a
"reorder response" of a network, a concept somewhat similar to the
"system response" or "impulse response" used in traditional system
theory. For a subnet under stationary conditions, RD is the
probability density of the packet displacement. RD measured on
individual subnets can be combined, using the convolution operation,
to predict the end-to-end reorder characteristics of the network
formed by the cascade of subnets under a fairly broad set of

conditions [Pi05b]. RD also shows significant promise as a tool for
analytical modeling of reordering, as demonstrated with a load-
balancing scenario in [Pi06]. Use of a threshold to define the
condition under which a packet is considered lost makes the metric
robust, efficient, and adaptable for different network and stream
characteristics.

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 [Ban02]. The
occupancy of this buffer, after each arrival, is used as the measure
of reordering. A threshold, used to declare a packet as lost, keeps
the complexity of computation within bounds. The threshold may be
selected based on application requirements in situations where the
late arrival of a packet makes it useless, e.g., a real-time
application. In [Ban02], this metric was called RD and buffer
occupancy was known as displacement.

RD and RBD are simple, yet useful, metrics for measurement and
evaluation of reordering. These metrics are robust against many
peculiarities, such as those discussed previously, and have a
computational complexity of O(N), where N is the received sequence
size. RD is orthogonal to loss and duplication, whereas RBD is
orthogonal to duplication.

A detailed comparison of these and other proposed metrics for
reordering is presented in [Pi07].

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

3.1. Receive Index (RI)

Consider a sequence of packets (1, 2, ..., N) transmitted over a
network. A receive index RI (1, 2, ...), is a value assigned to a
packet as it arrives at its destination, according to the order of
arrival. 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 6.) In the absence of reordering, the sequence
number of the packet and the receive index are the 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.

3.2. Out-of-Order Packet

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

3.3. Displacement (D)

   Displacement (D) 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 indicates the
   earliness of a packet and a positive displacement 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

3.4. Displacement Threshold (DT)

   The displacement threshold is a threshold on the displacement of
   packets 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
   definitely be classified as lost; the packet may 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 the 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.

If the DT selected is 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 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 5.

3.5. Displacement Frequency (FD)

Displacement Frequency FD[k] is the number of arrived packets having
a displacement of k, where k takes values from -DT to DT.

3.6. Reorder Density (RD)

RD is defined as the distribution of the Displacement Frequencies
FD[k], normalized with respect to N', where N' is the length of the
received sequence, ignoring lost and duplicate packets. N' is equal
to the sum(FD[k]) for k in [-DT, DT].

3.7. Expected Packet (E)

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

3.8. 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 instant, the buffer occupancy is equal to the number of
out-of-order packets in the buffer, including the newly arrived
packet. One buffer location is assumed for each packet, although it
is possible to extend the concept to the case where the number of
bytes is used for buffer occupancy. For example, consider 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.

3.9. 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
with the case of DT for RD, BT is used for loss and duplication
classification for Reorder Buffer-occupancy Density (RBD) computation
(see Section 3.11). BT provides robustness and limits the
computational complexity of RBD.

3.10. 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 arrival instances after which the occupancy takes
the value of k.

3.11. 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].

4. 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 an offset dm", i.e.,

        RI = m + dm; D = 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 (RI)  1       2       3       4       5       6
Displacement (D)    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 (RI)  1       2       3       5       6       -
Displacement (D)    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. That is:

$$S[k] = \{r(m, dm) \mid dm = k\}$$

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 duplicate or lost packets.

$$RD[k] = |S[k]| / N' \text{ for k not equal to zero}$$

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

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

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

5. 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 that introduce practical limits beyond which the choice of certain values becomes irrelevant. 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 maximum transmit and receive window sizes impose
a natural limit on the useful value of DT. Sequence number
wraparound may provide a useful upper bound for DT in some instances.

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. Any metric that
does not rely on a threshold to declare a packet as lost implicitly
makes one of two assumptions: a) A missing packet is not considered
lost until the end of the sequence, or b) the packet is considered
lost until it arrives. The former corresponds to the case where DT
is set to the length of the sequence. The latter leads to many
problems related to complexity and robustness.

6. Detection of Lost and Duplicate Packets

In RD, a packet is considered lost if it is late beyond DT.
Non-duplicate arriving packets do not have a copy in the buffer and
do not have a sequence number less (earlier) than E. In RBD, a
packet is considered lost if the buffer is filled to its threshold
BT. A packet is considered a duplicate when the sequence number is
less than the expected packet, or if the sequence number is already
in the buffer.

Since RI skips the sequence number of a lost packet, the question
arises as to how to assign an RI to subsequent packets that arrive
before it is known that the packet is lost. This problem arises only
when reorder metrics are calculated in real-time for an incoming
sequence, and not with offline computations. This concern 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 are
recomputed. The Go-back Method is only invoked when a packet is lost
and recomputing RD involves at most DT packets.

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
computation lags the arriving packet by DT. The lag associated with
the Stay-back Method is incurred only when a packet is missing.

   Another issue related to a metric and its implementation is the
   robustness against peculiarities that may occur in a sequence as
   discussed in Section 2. Consider, for example, the arrival sequence
   (1, 5430, 2, 3, 4, 5, ...). With RD, a sense of proportionality is
   easily maintained using the concept of threshold (DT), which limits
   the effects a rogue packet can have on the measurement results. In
   this example, when the displacement is greater than DT, rogue packet
   5430 is discarded. In this way the impact due to the rogue packet is
   limited, at most, to DT packets, thus imposing a limit on the amount
   of error it can cause in the results. Note also that a threshold
   different from DT can be used for the same purpose. For example, a
   pre-specified threshold that limits the time a packet remains in the
   buffer can make RBD robust against rogue packets.

7. Algorithms to Evaluate RD and RBD

   The algorithms to compute RD and RBD are given below. These
   algorithms are applicable for online computation of an incoming
   packet stream and provide an up-to-date metric for the packet stream
   read so far. For simplicity, the sequence numbers are considered to
   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. The algorithm for the Go-back Method is given in
   [Bar04]. Perl scripts for these algorithms are posted in [Per04].

7.1. Algorithm for RD

   Variables used:

    RI: receive index.
    S: Arrival under consideration for lateness/earliness computation.
    D: Lateness or earliness of the packet being processed: dm for m.
    FD[-DT..DT]: Frequency of lateness and earliness.
    window[1..DT+1]: List of incoming sequence numbers; FIFO buffer.
    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 (until window is empty):

      If (window or buffer contains sequence number RI)
      {
         Move sequence number out of window to S # window is FIFO

```
        D = RI - S; # compute displacement

        If (absolute(D) <= DT) # Apply threshold
        {
           FD[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;
           # Note, an early arrival in window is moved to buffer if
           # its displacement is less or equal to DT.  Therefore, the
           # contents in buffer will have only possible RIs.  Thus,
           # clearing an RI as it is consumed prevents memory leaks
           # in buffer
        }
        # 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 FD 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;
    }
```

7.2. Algorithm for RBD

```
    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 If (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 obtain RBD[i], for all values of i using

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

8. 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
          FD[D]     1     1     1     1     1     2    3     4
          ---------------------------------------------------------
          (S, RI,D and FD[D] as described in Section 7.1)
          ---------------------------------------------------------
```

The last row (FD[D]) represents the current frequency of occurrence
of the displacement D, e.g., column 3 indicates FD[1] = 1 while
column 4 indicates FD[-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
          FD[D]       1        1       4      1       1
          RD[D]    0.125    0.125    0.5   0.125   0.125
          ------------------------------------------------
          (D,FD[D] and RD[D] as described in Section 7.1)
          ------------------------------------------------
```

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 7.2)
          -------------------------------------------------------------
```
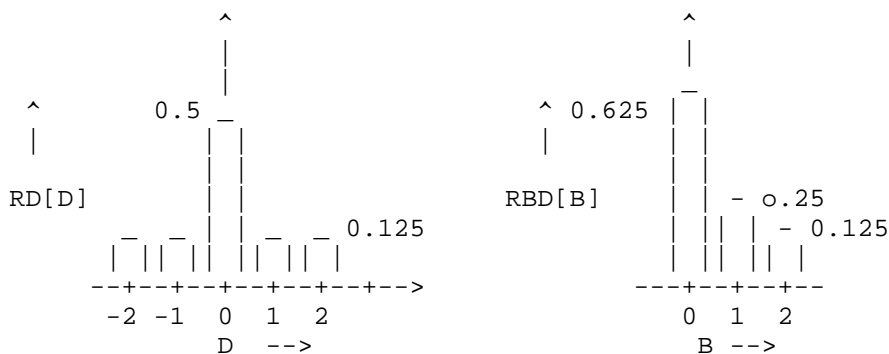
```
         ----------------------------------------------------------
         Table 4: Reorder Buffer-occupancy Density
         ----------------------------------------------------------
         B            0          1      2
         FB[B]        5          2      1
         RBD[B]     0.625     0.25   0.125
         ----------------------------------------------------------
         (B,FB[B] and RBD[B] as discussed in Section 7.2)
         ----------------------------------------------------------
```

Graphical representations of the densities are as follows:

```
                     ^                              ^
                     |                              |
                     |                             _
          ^       0.5 _                   ^ 0.625 | |
          |          | |                  |       | |
          |          | |                  |       | |
       RD[D]         | |                RBD[B]    | | - o.25
              _  _  | | _  _  0.125               | || | - 0.125
             | || || || || |                      | || || |
           --+--+--+--+--+--+-->                 ---+--+--+--
            -2 -1  0  1  2                          0  1  2
                  D  -->                              B  -->
```

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 FD[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
         FD[D]      1      2      3      4      5      6
         --------------------------------------------------------
         (S,RI,D and FD[D] as described in Section 7.1)
         --------------------------------------------------------
```
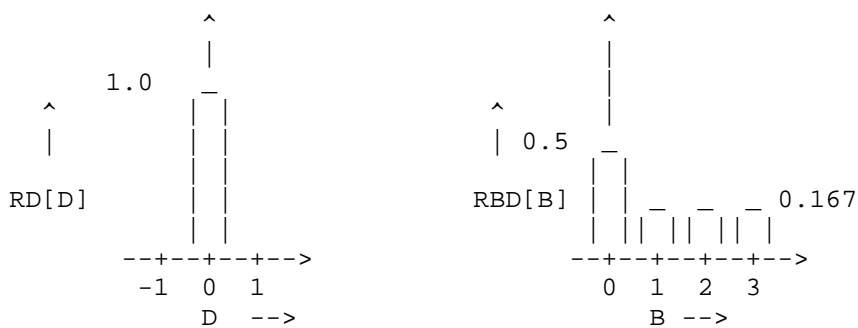
Table 6 illustrates the FB[B] for the above arrival sequence.

```
---------------------------------------------------
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 7.2)
---------------------------------------------------
```

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

```
               ^                              ^
               |                              |
        1.0   _                               |
     ^        | |                  ^          |
     |        | |                  | 0.5     _
     |        | |                  |        | |
 RD[D]        | |              RBD[B] | |    _   _   _ 0.167
     |        | |                  | || || || |
     --+--+--+-->                  --+--+--+--+-->
      -1  0  1                       0  1  2  3
         D  -->                        B  -->
```

c. Scenario with duplicate packets

Consider a sequence of 6 packets (1, 3, 2, 3, 4, 5) with DT = 2.
Table 7 shows the computational steps when the RD algorithm is
applied to the above sequence to obtain FD[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
FD[D]     1     1     1     -     2     3
-------------------------------------------------------
(S, RI,D and FD[D] as described in Section 7.1)
-------------------------------------------------------
```

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 7.2)
---------------------------------------------------------
```

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

```
                      ^                             ^
                      |                             |
        ^             |                  ^    0.8   _
        |      0.6    _                  |          | |
        |             | |                |          | |
    RD[D]             | |            RBD[B]          | |
        0.2   _       | |   _  0.2                   | |   _  0.2
              | |     | |   | |                      | |   | |
        --+--+--+--+--+--+-->                    ---+--+--+--
        -2 -1  0  1  2                              0  1  2
               D  -->                               B  -->
```
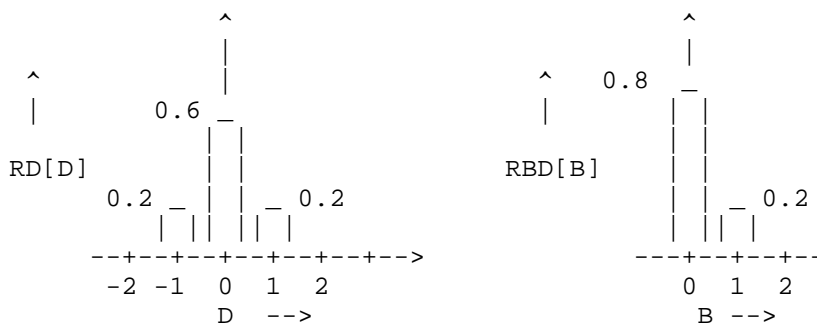
9. Characteristics Derivable from RD and RBD

   Additional information may be extracted from RD and RBD depending on
   the specific applications. For example, in the case of resource
   allocation at a node to recover from reordering, the mean and
   variance of buffer occupancy can be derived from RBD. For example:

   Mean occupancy of recovery buffer = sum(i*RBD[i] for 0 <= i <= BT)

   The basic definition of RBD may be modified to count the buffer
   occupancy in bytes as opposed to packets when the actual buffer space
   is more important. Another alternative is to use time to update the
   buffer occupancy compared to updating it at every arrival instant.

   The parameters that can be extracted from RD include the percentage
   of late (or early) packets, mean displacement of packets, and mean
   displacement of late (or early) packets [Ye06]. For example, the
   fraction of packets that arrive after three or more of their
   successors according to the order of transmission is given by Sum

   [RD[i] for 3<=i<=DT]. RD also allows for extraction of parameters
   such as entropy of the reordered sequence, a measure of disorder in
   the sequence [Ye06]. Due to the probability mass function nature of
   RD, it is also a convenient measure for theoretical modeling and
   analysis of reordering, e.g., see [Pi06].

10. Comparison with Other Metrics

   RD and RBD are compared to other metrics of [RFC4737] in [Pi07].

11. Security Considerations

   The security considerations listed in [RFC4737], [RFC3763], and
   [RFC4656] are extensive and directly applicable to the usage of these
   metrics; thus, they should be consulted for additional details.

12. References

12.1. Normative References

   [RFC2330] Paxson, V., Almes, G., Mahdavi, J., and M. Mathis,
              "Framework for IP Performance Metrics", RFC 2330, May
              1998.

   [Pi07]   N. M. Piratla and A. P. Jayasumana, "Metrics for Packet
              Reordering - A Comparative Analysis," International
              Journal of Communication Systems (IJCS), Vol. 21/1, 2008,
              pp: 99-113.

12.2. Informative References

   [Ben99]   J. C. R. Bennett, C. Partridge and N. Shectman, "Packet
              Reordering is Not Pathological Network Behavior," IEEE/ACM
              Trans. on Networking , Dec. 1999, pp.789-798.

   [Jai03]   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.

   [Pax97]   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.

   [Boh03]   S. Bohacek, J. Hespanha, J. Lee, C. Lim and K.Obraczka,
              "TCP-PR: TCP for Persistent Packet Reordering," Proc. of
              the IEEE 23rdICDCS, May 2003, pp.222-231.

   [Bla02]   E. Blanton and M. Allman, "On Making TCP More Robust to
              Packet Reordering," ACM Computer Comm. Review, 32(1), Jan.
              2002, pp.20-30.

   [Lao02]   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.

   [Bar04]   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.

   [Ban02]   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, pp. 332-342.

   [Pi05a]   N. M. Piratla, "A Theoretical Foundation, Metrics and
              Modeling of Packet Reordering and Methodology of Delay
              Modeling using Inter-packet Gaps," Ph.D. Dissertation,
              Department of Electrical and Computer Engineering,
              Colorado State University, Fort Collins, CO, Fall 2005.

   [Pi05b]   N. M. Piratla, A. P. Jayasumana and A. A. Bare, "RD: A
              Formal, Comprehensive Metric for Packet Reordering," Proc.
              5th International IFIP-TC6 Networking Conference
              (Networking 2005), Waterloo, Canada, May 2-6, 2005, LNCS
              3462, pp: 78-89.

   [Pi06]   N. M. Piratla and A. P. Jayasumana, "Reordering of Packets
              due to Multipath Forwarding - An Analysis," Proc. IEE
              Intl. Conf. Communications ICC 2006, Istanbul, Turkey,
              Jun. 2006, pp:829-834.

   [Per04]   Perl Scripts for RLED and RBD,
              http://www.cnrl.colostate.edu/packet_reorder.html, Last
              modified on Jul. 18, 2004.

   [Ye06]   B. Ye, A. P. Jayasumana and N. Piratla, "On Monitoring of
              End-to-End Packet Reordering over the Internet," Proc.
              Int. Conf. on Networking and Services (ICNS'06), Santa
              Clara, CA, July 2006.

   [RFC4737] Morton, A., Ciavattone, L., Ramachandran, G., Shalunov,
              S., and J. Perser, "Packet Reordering Metrics", RFC 4737,
              November 2006.

   [RFC3763] Shalunov, S. and B. Teitelbaum, "One-way Active
              Measurement Protocol (OWAMP) Requirements", RFC 3763,
              April 2004.

   [RFC4656] Shalunov, S., Teitelbaum, B., Karp, A., Boote, J., and M.
              Zekauskas, "A One-way Active Measurement Protocol
              (OWAMP)", RFC 4656, September 2006.

   [Pi05c]   N. M. Piratla, A. P. Jayasumana and T. Banka, "On Reorder
              Density and its Application to Characterization of Packet
              Reordering," Proc. 30th IEEE Local Computer Networks
              Conference (LCN 2005), Sydney, Australia, Nov. 2005,
              pp:156-165.

13. Contributors

   Jerry McCollom
   Hewlett Packard, 3404 East Harmony Road
   Fort Collins, CO 80528, USA

   EMail: jerry_mccollom@hp.com

Authors' Addresses

   Anura P. Jayasumana
   Computer Networking Research Laboratory
   Department of Electrical and Computer Engineering
   1373 Colorado State University,
   Fort Collins, CO 80523, USA

   EMail: Anura.Jayasumana@colostate.edu

   Nischal M. Piratla
   Deutsche Telekom Laboratories
   Ernst-Reuter-Platz 7
   D-10587 Berlin, Germany

   EMail: Nischal.Piratla@telekom.de

   Tarun Banka
   Computer Networking Research Laboratory
   Department of Electrical and Computer Engineering
   1373 Colorado State University
   Fort Collins, CO 80523, USA

   EMail: Tarun.Banka@colostate.edu

   Abhijit A. Bare
   Agilent Technologies, Inc.
   900 South Taft Ave.
   Loveland, CO 80537, USA

   EMail: abhijit_bare@agilent.com

   Rick Whitner
   Agilent Technologies, Inc.
   900 South Taft Ave.
   Loveland, CO 80537, USA

   EMail: rick_whitner@agilent.com