Network Working Group   A.Morton
Internet Draft   L.Ciavattone
Document: <draft-ietf-ippm-reordering-02.txt>   G.Ramachandran
Category: Standards Track   AT&T Labs
                                                     S.Shalunov
                                                      Internet2
                                                       J.Perser
                                                        Spirent

Packet Reordering Metric for IPPM

Status of this Memo

   Abstract

   This memo defines a simple metric to determine if a network has
   maintained packet order. It provides motivations for the new metric,
   suggests a metric definition, and discusses the issues associated
   with measurement. The memo includes sample metrics to quantify the
   extent of reordering in several useful dimensions. Some examples of
   evaluation using the various sample metrics are included.

1. Conventions used in this document

   The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT",
   "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this
   document are to be interpreted as described in RFC 2119 [2].
   Although RFC 2119 was written with protocols in mind, the key words
   are used in this document for similar reasons. They are used to
   ensure the results of measurements from two different
   implementations are comparable, and to note instances when an
   implementation could perturb the network.

2. **Introduction**

   Ordered delivery is a property of successful packet transfer
   attempts, where the packet sequence ascends for each arriving packet
   and there are no backward steps.

   An explicit sequence number, such as an incrementing message number
   or the packet sending time carried in each packet, establishes the
   Source Sequence.

   The presence of reordering at the Destination is based on arrival
   order.

   This metric is consistent with RFC 2330 [3], and classifies arriving
   packets with sequence numbers smaller than their predecessors as
   out-of-order, or reordered. For example, if arriving packets are
   numbered 1,2,4,5,3, then packet 3 is reordered. This is equivalent
   to Paxon's reordering definition in [4], where "late" packets were
   declared reordered. The alternative is to emphasize "premature"
   packets instead (4 and 5 in the example), but only the arrival of
   packet 3 distinguishes this circumstance from packet loss. Focusing
   attention on late packets allows us to maintain orthogonality with
   the packet loss metric. The metric's construction is very similar to
   the sequence space validation for received segments in RFC793 [5].
   Earlier work to define ordered delivery includes [6], [7], [8 and
   more ???.

2.1 **Motivation**

   A reordering metric is relevant for most applications, especially
   when assessing network support for Real-Time media streams. The
   extent of reordering may be sufficient to cause a received packet to
   be discarded by functions above the IP layer.

   Packet order is not expected to change during transfer, but several
   specific path characteristics can cause their order to change.

   Examples are:
   * When two paths, one with slightly longer transfer time, support a
     single packet stream or flow, then packets traversing the longer
     path may arrive out-of-order. Multiple paths may be used to
     achieve load balancing, or may arise from route instability.
   * To increase capacity, a network device designed with multiple
     processors serving a single port may reorder as a byproduct.
   * A layer 2 retransmission protocol that compensates for an error-
     prone link may cause packet reordering.

* If for any reason, the packets in a buffer are not serviced in the
        order of their arrival, their order will change.
      * If packets in a flow are assigned to multiple buffers (following
        evaluation of traffic characteristics, for example), and the

        buffers have different occupations and/or service rates, then
        order will likely change.

   The ability to restore order at the destination will likely have
   finite limits. Practical hosts have receiver buffers with finite
   size in terms of packets, bytes, or time (such as de-jitter
   buffers). Once the initial determination of reordering is made, it
   is useful to quantify the extent of reordering, or lateness, in all
   meaningful dimensions.

2.2 Goals and Objectives

   The definitions below intend to satisfy the goals of:
     1. Determining whether or not packet order is maintained.
     2. Quantifying the extent (achieving this second goal requires
        assumptions of upper layer functions and capabilities to
        restore order, and therefore several solutions).

   Reordering Metrics MUST:

   + be relevant to one or more known applications
   + be computable "on the fly"
   + work with Poisson and Periodic test streams
   + work even if the stream has duplicate or lost packets

   Reordering Metrics SHOULD:

   + have concatenating results for segments measured separately
   + have simplicity for easy consumption and understanding
   + have relevance to TCP performance
   + have relevance to Real-time application performance

3. An Ordered Arrival Singleton Metric

   The IPPM framework RFC 2330 [3] gives the definitions of singletons,
   samples, and statistics.

   The evaluation of packet order requires several supporting concepts.
   The first is a sequence number applied to packets at the source to
   uniquely identify the order of packet transmission. The sequence
   number may be established by a simple message number, a byte stream
   number, or it may be the actual time when each packet departs from

the Src.

The second supporting concept is a stored value which is the "next
expected" packet number. Under normal conditions, the value of Next
Expected (NextExp) is the sequence number of the previous packet
(plus 1 for message numbering). In byte stream numbering, NextExp
is a value 1 byte greater than the last in-order packet sequence
number + payload. If Src time is used as the sequence number,

NextExp is the Src time from the last in-order packet + 1 clock
tick.

Each packet within a packet stream can be evaluated for its order
singleton metric.

3.1 Metric Name:

Type-P-Non-Reversing-Order

3.2 Metric Parameters:

+ Src, the IP address of a host

+ Dst, the IP address of a host

+ SrcTime, the time of packet emission from the Src (or wire time)

+ s, the packet sequence number applied at the Src, in units of
   messages.

+ SrcByte, the packet sequence number applied at the Src, in units
   of payload bytes.

+ NextExp, the Next Expected Sequence number at the Dst, in units
   of messages, time, or bytes.

+ PayloadSize, the number of bytes contained in the information
   field and referred to when the SrcByte sequence is based on byte
   transfer.

3.3 Definition:

In-order packets have sequence numbers (or Src times) greater than
or equal to the value of Next Expected. Each new in-order packet
will increase the Next Expected (typically by 1 for message
numbering, or the payload size plus 1 for byte numbering). The Next
Expected value cannot decrease, thereby specifying non-reversing
order as the basis to identify reordered packets.

A reordered packet outcome occurs when a single IP packet at the Dst
Measurement Point results in the following:
The packet has a Src sequence number lower than the Next Expected
(NextExp), and therefore the packet is reordered. The Next Expected
value does not change on the arrival of this packet.

This definition can also be specified in pseudo-code.
On successful arrival of a packet with sequence number s:

```
     if s >= NextExp, /* s is in-order */
             then
             NextExp = s + PayloadSize + 1;
     else            /* when s < NextExp */
```

```
             designate packet s as reordered;
```

The sequence number s may be replaced by SrcTime or SrcByte. When
using s (message-based sequence numbering) or Src time,
PayloadSize=0.

3.4 Discussion

Any arriving packet bearing a sequence number from the sequence that
establishes the Next Expected value can be evaluated to determine if
it is in-order, or reordered, based on a previous packet's arrival.
In the case where Next Expected is Undefined (because the arriving
packet is the first successful transfer), the packet is designated
in-order.

If duplicate packets (multiple non-corrupt copies) arrive at the
destination, they MUST be noted and only the first to arrive is
considered for further analysis (additional copies would be declared
reordered according to the definition above). This requirement has
the same storage requirements as earlier IPPM metrics, and follows
the precedent of RFC 2679.

Packets with s > NextExp are a special case of in-order delivery.
This condition indicates a sequence discontinuity, either because of
packet loss or reordering. Reordered packets must arrive for the
sequence discontinuity to be part of a reordering event (defined in
the next section). Discontinuities are easiest to detect with
message numbering or payload byte numbering where payload size is
constant, and may be possible with Periodic Streams and Src Time
numbering.

4. Sample Metrics

In this section, we define metrics applicable to a sample of packets
from a single Src sequence number system. We begin with a simple

ratio metric indicating the reordered portion of the sample. When
this ratio is zero, no further reordering metrics are needed for
that sample.

When reordering occurs, it is highly desirable to assert the degree
to which a packet is out-of-order, or reordered with respect to a
sample of packets. This section defines several metrics that
quantify the extent of reordering in various units of measure. Each
"extent" metric highlights a relevant application.

4.1 Reordered Packet Ratio

4.1.1 Metric Name:

Type-P-Reordered-Ratio-Poisson/Periodic-Stream

4.1.2 Metric Parameters:

The parameter set includes Type-P-Non-Reversing-Order singleton
parameters, the parameters unique to Poisson or Periodic Streams,
plus the following:

+ T0, a start time

+ Tf, an end time

+ dT, a waiting time for each packet to arrive

4.1.3 Definition:

For the packets arriving successfully between T0 and Tf+dT, the
ratio of reordered packets in the sample is

(Total of Reordered packets) / (Total packets received)

This fraction may be expressed as a percentage (multiply by 100%).
Note that in the case of duplicate packets, only the first copy is
used.

4.2 Reordering Events and their Extent
Note: This section is expected to evolve further as we integrate the
various metrics of reordering extent (n-reordering and other
distance metrics used in earlier drafts). The co-authors are not yet
satisfied with all aspects of this section, and comments are
welcome.

4.2.1 Metric Name:

Type-P-packet-n-Reordering-Event-Poisson/Periodic-Stream

4.2.2 Parameter Notation:

Let n be a positive integer (a parameter). Let k be a positive
integer (sample size, the number of packets sent). Let l be a non-
negative integer representing the number of packets that were
received out of the k packets sent. (Note that there is no
relationship between k and l: on one hand, losses can make l less
than k; on the other hand, duplicates can make l greater than k.)
Assign each sent packet a sequence number, 1 to k.

Let s[1], s[2], ..., s[l] be the original sequence numbers of the
received packets, in the order of arrival.

4.2.3 Definition 1:

Received packet number i (n < i <= l), with Src sequence number s
(s[i]), is reordered to the extent n if and only if for all j such
that i-n <= j < i we have s[j] > s[i].

Claim: If by this definition, the extent of a packet's reordering is
n and 0 < n' < n, then the packet is also reordered to the n'
extent.

Note: This definition is illustrated by C code in Appendix A. It
determines the presence of n-reordering events for a particular
value of n (when actually writing applications that would report the
metric, one would probably report it for several values of n, such
as 1, 2, 3, 4 -- and maybe a few more consecutive values).
Also, this definition does not assign an extent to all reordered
packets as defined by the singleton metric, in particular when
blocks of successive packets are reordered. (In the arrival sequence
s={1,2,3,7,8,9,4,5,6}, packets 4, 5, and 6 are reordered, but only 4
is assigned a reordering extent according to Definition 1.) All
reordered packets are assigned a reordering extent by associating
them with a specific reordering event, as defined below.

4.2.4 Definition 2:
Note: The intent of this section is to assign a reordering extent to
all reordered packets, not just the ones identified by Definition 1.
This definition is new in this version and needs more study.

A packet s[i] that satisfies Definition 1 constitutes an n-
Reordering Event with the following characteristics:

1. The maximum value of n that satisfies Definition 1 is the extent
of the reordering event. (Extent n is assigned to all packets

associated with this event in part 3 below.)

2. The in-order packet arrival defined as beginning the event
(having indicated a sequence discontinuity) is s[j] for j that
satisfies the following:

$$\min\{j|1<=j<i\} \text{ for which } s[j]> s[i]$$

Typically i-n=j. This aspect of a reordering event is used later in
the definition of the gap between successive events.

3. The arrival of any subsequent reordered packets with sequence
number s meeting the following condition:

$$s[j] > s[*] > s[i], \text{ or}$$
(s at beginning of event) > s > (lowest s in the reordering event)

are associated with the reordering event first identified by s[i],
the sequence discontinuity that starts the event at s[j], and are
assigned the same reordering extent, n.
>>>
Comment on Part 3.: For some arrival orders, the assignment of the
same extent to all packets in a reordering event will tend to
underestimate their true extent. However, a pure position/distance
metric (as appeared in earlier versions of this draft) would tend to

overestimate the receiver storage needed. We need to weigh the value
of adding more complexity in this definition against the accuracy it
would provide.
A more accurate and complex procedure would be to count any
additional in-order packets that arrive after a reordering event is
identified, and add them to the extent of the first reordered packet
(up to some counter limit of interest) for packets in the interval
s[i] < s[*] < s[j].
Those who desire "on-the-fly" calculation must assess whether such a
procedure is feasible.

Discussion:

A receiver must perform some amount of "work" to restore order to
all packets that are part of an n-reordering event. The extent n
gives the number of packets that must be held in the receiver's
buffer while waiting for the reordered packets in the sequence. As
reordered packets arrive, the amount of work stays the same if they
are all part of the same reordering event. If new reordering events
occur, the work in terms of buffer size can increase. See Examples
section (specific example to be provided).

Knowledge of the reordering extent n is particularly useful for

determining the portion of reordered packets that can or cannot be restored to order in a typical TCP receiver buffer based on their arrival order alone (and without the aid of retransmission).

Important special cases are n=1 and n=3:

- For n=1, absence of 1-reordering events means the sequence numbers that the receiver sees are monotonically increasing with respect to the previous arriving packet.

- For n=3, a NewReno TCP sender would retransmit 1 packet in response to a 3-reordering event and therefore consider this a loss event for the purposes of congestion control (the sender will half its congestion window). Detecting 3-reordering events is useful for determining the portion of reordered packets that are in fact as good as lost.

We note that the definition of n-reordering events cannot predict the exact number of packets unnecessarily retransmitted by a TCP sender under some circumstances, such as closely-spaced reordering events. The definition is less complicated than a TCP implementation where both time and position influence the sender's behavior.

A sample's reordering extents may be expressed as a histogram, to easily summarize the frequency of various extents.

4.3 Reordering Offset

Any packet whose sequence number causes the Next Expected value to increment by more than the usual increment indicates a discontinuity in the sequence. From this point on, any reordered packets can be assigned Offset values indicating the storage in bytes and lateness in terms of buffer time that a receiver must possess to accommodate them. The various Offset metrics are calculated only on reordered packets, as identified by the ordered arrival singleton in section 3.

4.3.1 Metric Name: Type-P-packet-Late-Time-Poisson/Periodic-Stream

Metric Parameters: In addition to the parameters defined for Type-P-Non-Reversing-Order, we specify:

+ DstTime, the time that each packet in the stream arrives at Dst

Definition: Lateness in time is calculated using Dst times. When packet i is reordered, and part of a reordering event with n extent (assuming j=i-n):

```
LateTime(i) = DstTime(i)-DstTime(i-n)
```

Alternatively, using similar notation to that of <u>section 4.2</u>, an
equivalent definition is:
```
LateTime(i) = DstTime[i]-DstTime[j], for min{j|1<=j<i} that
satisfies s[j]>s[i], or SrcTime[j]>SrcTime[i].
```

<u>4.3.2</u> **Metric Name: Type-P-packet-Byte-Offset-Poisson/Periodic-Stream**

   Metric Parameters: We use the same parameters defined above.

   Definition: Byte stream offset is the sum of the payload sizes of
   all intervening packets between the reordered packet and the
   discontinuity (including the packet at the discontinuity).

   For reordered packet i
```
ByteOffset(i) = Sum[in-order packets to start of the reordering
event]
              = Sum[PayloadSize(packet at i-1),
                    PayloadSize(packet at i-2), ...
                    PayloadSize(packet at i-n)]
```

   Alternatively, if all payload sizes are equal:
```
ByteOffset(i) = n * PayloadSize  where n is the reordering extent.
```
   >>>>Comment: Previous comments on the accuracy of extent n apply
   here as well.

<u>4.3.3</u> **Discussion**

   The Offset metrics can predict whether reordered packets will be
   useful in a general receiver buffer system with finite limits. The
   limit may be the number of bytes or packets the buffer can store, or

   the time of storage prior to a cyclic play-out instant (as with de-
   jitter buffers).

   Note that the One-way IPDV [9] gives the delay variation for a
   packet w.r.t. the preceding packet in the source sequence. Lateness
   and IPDV give an indication of whether a buffer at Dst has
   sufficient storage to accommodate the network's behavior and restore
   order. When an earlier packet in the Src sequence is lost, IPDV will
   necessarily be undefined for adjacent packets, and Late Time may
   provide the only way to evaluate the usefulness of a packet.

   In the case of de-jitter buffers, there are circumstances where the
   receiver employs loss concealment at the intended play-out time of a
   late packet. However, if this packet arrives out of order, the Late
   Time determines whether the packet is still useful. IPDV no longer

applies, because the receiver establishes a new play-out schedule
with additional buffer delay to accommodate similar events in the
future - this requires very minimal processing.

When packets in the stream have variable sizes, it may be most
useful to characterize Offset in terms of the payload size(s) of
stored packets (using byte stream numbering).

4.4 Gaps between multiple Reordering Events

4.4.1 Metric Name:

Type-P-packet-Reordering-Event-Gap-Poisson/Periodic-Stream

4.4.2 Parameters:

No new parameters.

4.4.3 Definition:

A reordering event with extent n is detected according to section
4.2 with the arrival of packet s[i]. The next reordering event with
extent n' is detected at packet i', and there are no reordering
events between i and i'.

The Reordering Event Gap is the difference between the arrival
positions the packets, as shown below (assuming j=i-n):

Gap(i) = (i'-n') - (i-n)

Gaps may also be expressed in time:

GapTime(i) = DstTime(i'-n') - DstTime(i-n)

The Gaps between a sample's reordering events may be expressed as a
histogram, to easily summarize the frequency of various extents.

4.4.4 Discussion

When separate reordering events can be distinguished, then an event
count may also be reported (along with the event description, such
as the number of reordered packets and their extents or offsets).
The distribution of various metrics may also be reported and
summarized by the mode, average, range, histogram, etc.

The Gap metric may help to correlate the frequency of reordering

events with their cause.

5. Measurement Issues

The results of tests will be dependent on the time interval between
measurement packets (both at the Src, and during transport where
spacing may change). Clearly, packets launched infrequently (e.g.,
1 per 10 seconds) are unlikely to be reordered.

Test streams may prefer to use a periodic sending interval so that a
known temporal bias is maintained, also bringing simplified results
analysis (as described in RFC 3432 [10]). In this case, the periodic
sending interval should be chosen to reproduce the closest Src
packet spacing expected.
<<<<Ed.Note: Need to expand this further, it is a very important
consideration.

The Non-reversing order criterion and all metrics described above
remain valid and useful when a stream of packets experiences packet
loss, or both loss and reordering. In other words, losses alone do
not cause subsequent packets to be declared reordered.

Assuming that the necessary sequence information (sequence number
and/or source time stamp) is included in the packet payload
(possibly in application headers such as RTP), packet sequence may
be evaluated in a passive measurement arrangement. Also, it is
possible to evaluate sequence at a single point along a path, since
the usual need for synchronized Src and Dst Clocks may be relaxed to
some extent.

When the Src sequence is based on byte stream, or payload numbering,
care must be taken to avoid declaring retransmitted packets
reordered. The additional reference of Src Time is one way to avoid
this ambiguity.

Since this metric definition may use sequence numbers with finite
range, it is possible that the sequence numbers could reach end-of-
range and roll over to zero during a measurement. By definition,
the Next Expected value cannot decrease, and all packets received
after a roll-over would be declared reordered. Sequence number

roll-over can be avoided by using combinations of counter size and
test duration where roll-over is impossible (and sequence is reset
to zero at the start). Also, message-based numbering results in
slower sequence consumption. There may still be cases where
methodological mitigation of this problem is desirable (e.g., long-
term testing). The elements of mitigation are:

1. There must be a test to detect if a roll-over has occurred. It

would be nearly impossible for the sequence numbers of successive
packets to jump by more than half the total range, so these large
discontinuities are designated as roll-over.

2. All sequence numbers used in computations are represented in a
sufficiently large precision. The numbers have a correction applied
(equivalent to adding a significant digit) whenever roll-over is
detected.

3. Reordered packets coincident with sequence numbers reaching end-
of-range must also be detected for proper application of correction
factor.

6. Examples of Arrival Order Evaluation

This section provides some examples to illustrate how the non-
reversing order criterion works, and the value of viewing reordering
in both the dimensions of time and position.

Table 1 gives a simple case of reordering, where one packet (the
packet with s=4) arrives out-of-order. Packets are arranged
according to their arrival, and message numbering is used.

Table 1 Example with Packet 4 Reordered,
Sending order(SrcNum@Src): 1,2,3,4,5,6,7,8,9,10

s @Dst	NextExp	Src Time	Dst Time	Delay	IPDV	Dst Order	Byte Offset	Late Time
1	1	0	68	68		1		
2	2	20	88	68	0	2		
3	3	40	108	68	0	3		
5	4	80	148	68	-82	4		
6	6	100	168	68	0	5		
7	7	120	188	68	0	6		
8	8	140	208	68	0	7		
4	9	60	210	150	82	8	400	62
9	9	160	228	68	0	9		
10	10	180	248	68	0	10		

Each column gives the following information:

S   Packet sequence number at the Source.
NextExp The value of NextExp when the packet arrived(before
update).

SrcTime Packet time stamp at the Source, ms.
DstTime Packet time stamp at the Destination, ms.
Delay   1-way delay of the packet, ms.
IPDV   IP Packet Delay Variation, ms

```
        IPDV = Delay(SrcNum)-Delay(SrcNum-1)
DstOrder Order in which the packet arrived at the Destination.
Byte Offset  The Byte Offset of a reordered packet, in bytes.
LateTime The lateness of a reordered packet, in ms.
```

We can see that when packet 4 arrives, NextExp=9, and it is declared reordered. We compute the extent of the reordering event as follows:

Using the notation <s[1], ..., s[i], ..., s[l]>, the received packets are represented as:

```
                          \/
s = 1, 2, 3, 5, 6, 7, 8, 4, 9, 10
i = 1, 2, 3, 4, 5, 6, 7, 8, 9, 10
                       /\
```

```
when n=1, 7<=J<8, and 8 > 4, so the reordering extent is 1 or more.
when n=2, 6<=J<8, and 7 > 4, so the reordering extent is 2 or more.
when n=3, 5<=J<8, and 6 > 4, so the reordering extent is 3 or more.
when n=4, 4<=J<8, and 5 > 4, so the reordering extent is 4 or more.
when n=5, 3<=J<8, but 3 < 4, and 4 is the maximum extent.
```

Further, we can compute the Late Time (210-148=62ms using DstTime) compared to packet 5's arrival. If Dst has a de-jitter buffer that holds more than 4 packets, or at least 62 ms storage, packet 4 may be useful. Note that 1-way delay and IPDV also indicate unusual behavior for packet 4.

If all packets contained 100 byte payloads, then Byte Offset is equal to 400 bytes.

Table 2 Example with Packets 5 and 6 Reordered,
Sending order(s @Src): 1,2,3,4,5,6,7,8,9,10

s @Dst	Src NextExp	Dst Time	Dst Time	Delay	IPDV	Dst Order	Byte Offset	Late Time
1	1	0	68	68		1		
2	2	20	88	68	0	2		
3	3	40	108	68	0	3		
4	4	60	128	68	0	4		
7	5	120	188	68	-22	5		
5	8	80	189	109	41	6	100	1
6	8	100	190	90	-19	7	100	2
8	8	140	208	68	0	8		
9	9	160	228	68	0	9		
10	10	180	248	68	0	10		

Table 2 shows a case where packets 5 and 6 arrive just behind packet
7, so both 5 and 6 are reordered. The Late times (189-188=1, 190-
188=2) are small.

Using the notation <s[1], ..., s[i], ..., s[l]>, the received
packets are represented as:

```
                \/ \/
s = 1, 2, 3, 4, 7, 5, 6, 8, 9, 10
i = 1, 2, 3, 4, 5, 6, 7, 8, 9, 10
                /\ /\
```

Considering packet 5[6] first:
when n=1, 5<=J<6, and 7 > 5, so the reordering extent is 1 or more.
when n=2, 4<=J<6, but 4 < 5, and 1 is the maximum extent.

Considering packet 6[7] next:
when n=1, 6<=J<7, and 5 < 6, so the packet at i=7 does not have its
own reordering extent, and must be part of the same reordering event
as packet 5[6]. Using the test of Section 4.2.4, Definition 2, we
find that the condition is met for packet 6[7]:

```
                 s[i] <  s   < s[i-n]
                 5[6] < 6[7] < 7[5]
```

A hypothetical sender/receiver pair may retransmit packet 5[8]
unnecessarily, since it is reordered with extent n=1(in agreement
with the singleton metric). However, the receiver cannot advance
packet 7[5] to the higher layers until after packet 6[7] arrives.
Therefore, the singleton metric correctly determined that 6[7] is
reordered, and both packets are part of a 1-reordering event.

Table 3 Example with Packets 4, 5, and 6 reordered
Sending order(s @Src): 1,2,3,4,5,6,7,8,9,10,11

s @Dst	NextExp	Src Time	Dst Time	Delay	IPDV	Dst Order	Byte Offset	Late Time
1	1	0	68	68		1		
2	2	20	88	68	0	2		
3	3	40	108	68	0	3		
7	4	120	188	68	-68	4		
8	8	140	208	68	0	5		
9	9	160	228	68	0	6		
10	10	180	248	68	0	7		
4	11	60	250	190	122	8	400	62
5	11	80	252	172	-18	9	400	64
6	11	100	256	156	-16	10	400	68
11	11	200	268	68	0	11		

The case in Table 3 is where three packets in sequence have long
transit times (packets with s = 4,5,and 6). Delay, Late time, and

Byte Offset capture this very well, and indicate variation in

reordering extent, while IPDV indicates that the spacing between
packets 4,5,and 6 has changed.

The histogram of Reordering extents (n) would be:

```
Bin          1  2  3  4  5  6  7
Frequency    0  0  0  3  0  0  0
```

Using the notation <s[1], ..., s[i], ..., s[l]>, the received
packets are represented as:

```
s = 1, 2, 3, 7, 8, 9,10, 4, 5, 6, 11
i = 1, 2, 3, 4, 5, 6, 7, 8, 9, 10,11
```

Considering packet 4[8] first:
when n=1, 7<=J<8, and 10> 4, so the reordering extent is 1 or more.
when n=2, 6<=J<8, and 9 > 4, so the reordering extent is 2 or more.
when n=3, 5<=J<8, and 8 > 4, so the reordering extent is 3 or more.
when n=4, 4<=J<8, and 7 > 4, so the reordering extent is 4 or more.
when n=5, 3<=J<8, but 3 < 4, and 4 is the maximum extent.

Considering packet 5[9] next:
when n=1, 8<=J<9, but 4 < 5, so the packet at i=9 does not have its
own reordering extent, and must be part of the same reordering event
as packet 4[8]. Using the test of Section 4.2.4, Definition 2, we
find that the condition is met for both packets 5[9] and 6[10]:

```
                s[i] <  s   < s[i-n]
                4[8] < 5[9] < 7[4]
                4[8] < 6[10]< 7[4]
```

This example shows again that the n-reordering event definition
identifies a single event (s=4) with a sufficient degree of
reordering to result in one unnecessary packet retransmission by the
New Reno TCP sender. Also, the reordered arrival of packets s=5 and
s=6 will allow the receiver process to pass packets 7 through 10 up
the protocol stack (the singleton metric indicates 5 and 6 are
reordered, and they are all part of one reordering event).

7. Security Considerations

7.1 Denial of Service Attacks

This metric requires a stream of packets sent from one host (Src) to
another host (Dst) through intervening networks. This method could

be abused for denial of service attacks directed at Dst and/or the
intervening network(s).

Administrators of Src, Dst, and the intervening network(s) should
establish bilateral or multi-lateral agreements regarding the

timing, size, and frequency of collection of sample metrics. Use of
this method in excess of the terms agreed between the participants
may be cause for immediate rejection or discard of packets or other
escalation procedures defined between the affected parties.

7.2 User data confidentiality

Active use of this method generates packets for a sample, rather
than taking samples based on user data, and does not threaten user
data confidentiality. Passive measurement must restrict attention to
the headers of interest. Since user payloads may be temporarily
stored for length analysis, suitable precautions MUST be taken to
keep this information safe and confidential.

7.3 Interference with the metric

It may be possible to identify that a certain packet or stream of
packets is part of a sample. With that knowledge at Dst and/or the
intervening networks, it is possible to change the processing of the
packets (e.g. increasing or decreasing delay) that may distort the
measured performance. It may also be possible to generate
additional packets that appear to be part of the sample metric.
These additional packets are likely to perturb the results of the
sample measurement.

To discourage the kind of interference mentioned above, packet
interference checks, such as cryptographic hash, may be used.

8. IANA Considerations

Since this metric does not define a protocol or well-known values,
there are no IANA considerations in this memo.

9. References

1 Bradner, S., "The Internet Standards Process -- Revision 3", BCP
   9, RFC 2026, October 1996.

2 Bradner, S., "Key words for use in RFCs to Indicate Requirement
   Levels", RFC 2119, March 1997.

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

    4 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.

    5 Postel, J., "Transmission Control Protocol", STD 7, RFC 793,
       September 1981.
       Obtain via: http://www.rfc-editor.org/rfc/rfc793.txt

    6 L.Ciavattone and A.Morton, "Out-of-Sequence Packet Parameter
       Definition (for Y.1540)", Contribution number T1A1.3/2000-047,
       October 30, 2000. ftp://ftp.t1.org/pub/t1a1/2000-A13/0a130470.doc

    7 J.C.R.Bennett, C.Partridge, and N.Shectman, "Packet Reordering is
       Not Pathological Network Behavior," IEEE/ACM Transactions on
       Neteworking, vol.7, no.6, pp.789-798, December 1999.

    8 D.Loguinov and H.Radha, "Measurement Study of Low-bitrate
       Internet Video Streaming"' Proceedings of the ACM SIGCOMM
       Internet Measurement Workshop 2001 November 1-2, 2001, San
       Francisco, USA.

    9 Demichelis, C., and Chimento, P., "IP Packet Delay Variation
       Metric for IP Performance Metrics (IPPM)", RFC 3393, November
       2002.

    10 Raisanen, V., Grotefeld, G., and Morton, A., "Network performance
       measurement with periodic streams", RFC 3432, November 2002.

11. Acknowledgments

The authors would like to acknowledge many helpful discussions with
Matt Mathis and Jon Bennett. We gratefully acknowledge the
foundation laid by the authors of the IP performance Framework [3].

12. Appendix A (informative)

Two example c-code implementations of reordering definitions follow:

Example 1 n-reordering ===

```
#include <stdio.h>

#define MAX_N   100

#define min(a, b) ((a) < (b)? (a): (b))
```

```
#define loop(x) ((x) >= 0? x: x + MAX_N)

/*
 * Read new sequence number and return it.  Return a sentinel value
of EOF
 * (at least once) when there are no more sequence numbers.  In this
example,
 * the sequence numbers come from stdin; in an actual test, they
would come
 * from the network.
 */
int
read_sequence_number()
```

```
{
        int             res, rc;
        rc = scanf("%d\n", &res);
        if (rc == 1) return res;
        else return EOF;
}


int
main()
{
        int             m[MAX_N];       /* We have m[j-1] == number
of
                                         * j-reordered packets. */
        int             ring[MAX_N];    /* Last sequence numbers
seen. */
        int             r = 0;          /* Ring pointer for next
write. */
        int             l = 0;          /* Number of sequence
numbers read. */
        int             s;              /* Last sequence number
read. */
        int             j;


        for (j = 0; j < MAX_N; j++) m[j] = 0;
        for (; (s = read_sequence_number()) != EOF; l++, r = (r+1) %
MAX_N) {
                for (j=0; j<min(l, MAX_N) && s<ring[loop(r-j-1)];
j++) m[j]++;
                ring[r] = s;
        }
        for (j = 0; j < MAX_N && m[j]; j++)
                printf("%d-reordering = %f%%\n", j+1, 100.0*m[j]/(l-
```

```
j-1));
        if (j == 0) printf("no reordering\n");
        else if (j < MAX_N) printf("no %d-reordering\n", j+1);
        else printf("only up to %d-reordering is handled\n", MAX_N);
        exit(0);
}


Example 2   singleton and n-reordering comparison =================

#include <stdio.h>

#define MAX_N   100
#define min(a, b) ((a) < (b)? (a): (b))
#define loop(x) ((x) >= 0? x: x + MAX_N)

/* Global counters */
int receive_packets=0;      /* number of recieved */
int reorder_packets=0;      /* number of reordered packets */
```

```
/* function to test if current packet has been reordered
 * returns 0 = not reordered
 *         1 = reordered
 */
int testorder1(int seqnum)   // Al
{
    static int NextExp = 1;
    int iReturn = 0;

    if (seqnum >= NextExp) {
            NextExp = seqnum+1;
    } else {
            iReturn = 1;
    }
    return iReturn;
}

int testorder2(int seqnum)   // Stanislav
{
        static int     ring[MAX_N];    /* Last sequence numbers
seen. */
        static int   r = 0;            /* Ring pointer for next write.
*/
        int            l = 0;          /* Number of sequence
numbers read. */
        int            j;
    int     iReturn = 0;

        l++;
```

```
    r = (r+1) % MAX_N;
        for (j=0; j<min(l, MAX_N) && seqnum<ring[loop(r-j-1)]; j++)
                iReturn = 1;
        ring[r] = seqnum;
    return iReturn;
}

int main(int argc, char *argv[])
{
        int i, packet;
    for (i=1; i< argc; i++) {
            receive_packets++;
            packet = atoi(argv[i]);
            reorder_packets += testorder2(packet);
    }
    printf("Received packets = %d, Reordered packets = %d\n",
receive_packets, reorder_packets);
    exit(0);
}
```

[13](#). **Author's Addresses**

    Al Morton
    AT&T Labs
    Room D3 - 3C06
    200 Laurel Ave. South
    Middletown, NJ 07748 USA
    Phone +1 732 420 1571 Fax +1 732 368 1192
    <acmorton@att.com>

    Len Ciavattone
    AT&T Labs
    Room C4 - 2B29
    200 Laurel Ave. South
    Middletown, NJ 07748 USA
    Phone +1 732 420 1239
    <lencia@att.com>

    Gomathi Ramachandran
    AT&T Labs
    Room C4 - 3D22
    200 Laurel Ave. South
    Middletown, NJ 07748 USA
    Phone +1 732 420 2353
    <gomathi@att.com>

    Stanislav Shalunov

Internet2
200 Business Park Drive, Suite 307
Armonk, NY 10504
Phone: + 1 914 765 1182
EMail: <shalunov@internet2.edu>

Jerry Perser
Spirent Communications
26750 Agoura Road
Calabasas, CA 91302 USA
Phone: + 1 818 676 2300
EMail: <jerry.perser@spirentcom.com>