Network Working Group
Internet Draft
Document: <draft-ietf-ippm-reordering-05.txt>
Category: Standards Track

A.Morton
L.Ciavattone
G.Ramachandran
AT&T Labs
S.Shalunov
Internet2
J.Perser
Consultant

Packet Reordering Metric for IPPM

Status of this Memo

This document is an Internet-Draft and is in full conformance with all provisions of Section 10 of RFC2026 [1].

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF), its areas, and its working groups. Note that other groups may also distribute working documents as Internet-Drafts. Internet-Drafts are draft documents valid for a maximum of six months and may be updated, replaced, or made obsolete by other documents at any time. It is inappropriate to use Internet-Drafts as reference material or to cite them other than as "work in progress."

The list of current Internet-Drafts can be accessed at http://www.ietf.org/ietf/1id-abstracts.txt

The list of Internet-Draft Shadow Directories can be accessed at http://www.ietf.org/shadow.html.

Abstract

This memo defines metrics to evaluate if a network has maintained packet order on a packet-by-packet basis. It provides motivations for the new metrics and discusses the measurement issues. The memo first defines a reordered singleton, and then uses it as the basis for sample metrics to quantify the extent of reordering in several useful dimensions. Additional metrics quantify the frequency of reordering and the distance between separate occurrences. We then define metrics with a receiver analysis orientation. Several examples of evaluation using the various sample metrics are included. An Appendix gives extended definitions for evaluating order with packet fragmentation.

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 detection of reordering at the Destination is based on packet arrival order in comparison with a non-reversing reference value.

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 sequentially numbered packets arrive 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], [9], [10] and [11.

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 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.

When one or more of the above path characteristics are present continuously, then reordering may be present on a steady-state basis. Measurements most easily detect this form of reordering when the spacing between packets is minimized. Transient reordering may occur in response to network instability; temporary routing loops can cause periods of extreme reordering.

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. A Reordered Packet Singleton Metric

The IPPM framework RFC 2330 [3] describes the notions of singletons, samples, and statistics. For easy reference:

> By a 'singleton' metric, we refer to metrics that are, in a sense, atomic. For example, a single instance of "bulk throughput capacity" from one host to another might be defined

> as a singleton metric, even though the instance involves measuring the timing of a number of Internet packets.

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 Source.

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).

Each packet within a packet stream can be evaluated with this order singleton metric.

3.1 Metric Name:

Type-P-Reordered

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 Source (or wire time)

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

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

+ NextExp, the Next Expected Sequence number at the Destination, 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:

The value of Type-P-Reordered is defined as false if s >= NextExp (the packet is in-order). In this case, NextExp is set to s+1.

The value of Type-P-Reordered is defined as true if s < NextExp (the packet is reordered). In this case, NextExp value does not change.

Since the Next Expected value cannot decrease, it provides a non-reversing order criterion to identify reordered packets.

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 + 1;
             Type-P-Reordered = False;
     else            /* when s < NextExp */
             Type-P-Reordered = True
```

We note that when s = NextExp, the original sequence has been maintained, and there is no discontinuity present.

3.4 Evaluation in Time or Byte Order

For the alternate sequence dimensions, in-order packets have byte stream numbers or Source times greater than or equal to the value of NextExp. Each new in-order packet will increase the NextExp SrcTime plus 1 clock tick when using Source times, or to SrcByte plus the payload size plus 1 for byte numbering. In the pseudo-code above, SrcByte (or SrcTime) replaces the sequence number s, and we have:

```
     if SrcByte >= NextExp, /* packet is in-order */
             then
             NextExp = SrcByte + PayloadSize + 1;
```

When using Source time, PayloadSize=0 (or a fixed time increment, if using a reliable periodic packet source).

3.5 Discussion

Any arriving packet bearing a sequence number from the sequence that

establishes the Next Expected value can be evaluated to determine whether 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.

This metric assumes re-assembly of packet fragments before evaluation. In principle, it is possible to use the Type-P-Reordered metric to evaluate reordering among packet fragments, but each fragment must contain source sequence information.
See the Appendix on fragment order evaluation for more detail.

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 (copies would be declared reordered according to the definition above). This requirement has the same

storage implications 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 defined as a reordering discontinuity (see next section). Discontinuities are easiest to detect with message numbering or payload byte numbering where payload size is constant (and retransmissions are distinguished), and may be possible with Periodic Streams and Source Time numbering.

4. Sample Metrics

In this section, we define metrics applicable to a sample of packets from a single Source 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 other packets. This section defines several metrics that quantify the extent of reordering in various units of measure. Each "extent" metric highlights a relevant use.

The metrics in the sub-sections below have a network characterization orientation, but also have relevance to receiver design.

4.1 Reordered Packet Ratio

4.1.1 Metric Name:

Type-P-Reordered-Ratio-Stream

4.1.2 Metric Parameters:

The parameter set includes Type-P-Reordered singleton parameters, the parameters unique to Poisson or Periodic Streams (as in RFC 2330 and RFC3432), 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 Extent

This section defines the extent to which packets are reordered, and associates a specific sequence discontinuity with each reordered packet.

4.2.1 Metric Name:

Type-P-packet-Reordering-Extent-Stream

4.2.2 Parameter Notation:

Given a stream of packets sent from a source to a destination, let K be the total number of packets in that stream.

Assign each packet a sequence number, a consecutive integer from 1 to K in the order of packet emission.

Let L be the total number of packets received out of the K packets

sent. Recall that identical copies (duplicates) have been removed, so L<=K.

Let s[1], s[2], ..., s[L], represent the original sequence numbers associated with the packets in order of arrival.

Consider a reordered packet (as identified in section 3) with arrival index i and source sequence number s[i]. There exists a set of indexes j (1<=j<i) such that s[j] > s[i].

4.2.3 Definition:

The reordering extent, e, of packet s[i] is defined to be i-j for the smallest value of j.

Informally, the reordering extent is the maximum distance, in packets, from a reordered packet to the earliest packet received that has a larger sequence number. If a packet is in-order, its reordering extent is undefined. The first packet to arrive is in-order by definition, and has undefined reordering extent.

>>>>>>> Comment on this definition of extent: For some arrival orders, the assignment of a simple position/distance as the reordering extent tends to overestimate the receiver storage needed

to restore order. 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 to calculate packet storage would be to subtract any earlier reordered packets that the receiver could pass on to the upper layers.
Those who desire "on-the-fly" calculation must assess whether such a procedure is feasible.

4.2.4 Discussion:

The packet with index j (s[j], identified in the Definition above) is the reordering discontinuity associated with packet with index i (s[i]). This definition is formalized below.

Note that the K packets in the stream could be some subset of a larger stream, but L is still the total number of packets received out of the K packets sent in that subset.

A receiver must possess storage to restore order to packets that are reordered. For cases with single reordered packets, the extent e gives the number of packets that must be held in the receiver's buffer while waiting for the reordered packet to complete the sequence. For more complex scenarios, the extent may be an overestimate of required storage. See Examples section (specific

example to be provided).

Knowledge of the reordering extent e is particularly useful for determining the portion of reordered packets that can or cannot be restored to order in a typical receiver buffer based on their arrival order alone (and without the aid of retransmission).

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

4.3 Reordering Offset

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-Stream

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

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

Definition: Lateness in time is calculated using Dst times. When received packet i is reordered, and has a reordering extent e, then:

LateTime(i) = DstTime(i)-DstTime(i-e)

Alternatively, using similar notation to that of section 4.2, 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].

4.3.2 Metric Name: Type-P-packet-Byte-Offset-Stream

Metric Parameters: We use the same parameters defined above.

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

For reordered packet i with a reordering extent e:
ByteOffset(i) = Sum[in-order packets back to reordering discon.]
= Sum[PayloadSize(packet at i-1 if in-order),
PayloadSize(packet at i-2 if in-order), ...

PayloadSize(packet at i-e if in-order)]

4.3.3 Discussion

The offset metrics can help 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 [12] 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 Discontinuities

4.4.1 Metric Name:

Type-P-packet-Reordering-Gap-Stream

4.4.2 Parameters:

No new parameters.

4.4.3 Definition of Reordering Discontinuity:

All reordered packets are associated with a packet at a reordering discontinuity, defined as the in-order packet arrival s[j] at the minimum value of j (1<=j<i) for which s[j]> s[i].

Recall that i - e = min(j). Subsequent reordered packets may be associated with the same s[j], or with a different discontinuity. This definition is used in the definition of the Reordering Gap, below.

4.4.4 Definition of Reordering Gap:

A reordering gap is the distance between successive reordering discontinuities. Type-P-packet-Reordering-Gap-Stream assigns a value to (all) packets in a stream.

If:

The packet s[j'] is found to be a reordering discontinuity, based on the arrival of reordered packet s[i'] with extent e', and

an earlier reordering discontinuity s[j], based on the arrival of reordered packet s[i] with extent e was already detected, and

i' > i, and

there are no reordering discontinuities between j and j',

then the Reordering Gap for packet s[j'] is the difference between the arrival positions the reordering discontinuities, as shown below:

Gap(j') = (j') - (j)

Otherwise:

The Type-P-packet-Reordering-Gap-Stream for the packet is 0.

Gaps may also be expressed in time:

GapTime(j') = DstTime(j') - DstTime(j)

4.4.5 Discussion

When separate reordering discontinuities can be distinguished, then a count may also be reported (along with the discontinuity description, such as the number of reordered packets associated with that discontinuity and their extents and offsets). The Gaps between a sample's reordering discontinuities may be expressed as a histogram, to easily summarize the frequency of various gaps. Reporting the mode, average, range, etc. may also summarize the

distributions.

The Gap metric may help to correlate the frequency of reordering discontinuities with their cause.

4.5 Reordering-free Runs

This section defines a metric based on a count of consecutive packets between reordered packets.

4.5.1 Metric Name:

Type-P-packet-Reordering-Free-Run-Stream

4.5.2 Parameters:

No new parameters.

4.5.3 Definition:

As packets in a sample arrive at the Destination, the count of packets to the next reordered packet is a Reordering-Free run. Note that the minimum run-length is one according to this definition. A pseudo code example follows:

```
r = 0; /* r is the run counter */
n = 0; /* n is the number of runs */
a = 0; /* a is the accumulator of in order packets */
p = 0; /* p is the number of packets */
q = 0; /* q is the squared sum of the run counters */

while(packets arrive with sequence number s)
{
     P++;
     if (s >= NextExp) /* s is in-order */
             then r++;
             a++;
     else    /* s is reordered */
             q+= r*r;
             r = 1;
```



```
             n++;
}
```

Each arrival of a reordered packet yields a new count in the Run vector. Long runs accompany periods where order was maintained, while short runs indicate frequent, or multi-packet reordering.

4.5.4 Discussion:

Each in-order arrival increments the run counter and the accumulator of in order packets, each reordered packet resets the run counter after adding it to the accumulator.

The percent of packets in order is 100*a/p

The average in order run length is a/n

The q counter gives an indication of variation of the in order runs from the average by comparing q/a to a/n ((q/a)/(a/n))

For example for 36 packets with 3 runs of 11 in-order packets we have:

```
p = 36
n = 3
a = 33
q = 3 * (11*11) = 363
ave io = 11
q/a = 11
(q/a)/ave = 1.0
```

For 36 packets with 3 runs, 2 of length 1 and one of length 31

```
p = 36
n = 3
a = 33
q = 1 + 1 + 961 = 963
ave io = 11
q/a = 29.18
(q/a)/ave = 2.65
```

5. Metric Related to Receiver Assessment

5.1 A TCP-Relevant Metric

5.1.1 Metric Name:

Type-P-packet-n-Reordering-Stream

5.1.2 Parameter Notation:

Let n be a positive integer (a parameter). Let k be a positive integer equal to the number of packets sent (sample size). 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, in order of packet emission.

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

5.1.3 Definitions

Definition 1: Received packet number i (n < i <= l), with source sequence number s[i], is n-reordered if and only if for all j such that i-n <= j < i, s[j] > s[i].

Claim: If by this definition, 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 n-reordering for a value of n=3 (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).
This definition does not assign an n 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 n-reordered, with n=3.)

Definition 2: The degree of n-reordering of the sample is m/l.

Definition 3: The degree of "monotonic reordering" of the sample is its degree of 1-reordering.

Definition 4: A sample is said to have no reordering if its degree of n-reordering is 0.

5.1.4 Discussion:

The degree of n-reordering may be expressed as a percentage, in which case the number from definition 2 is multiplied by 100.

Knowledge of n-reordering 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 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 an instance of 3-reordering and therefore consider this packet lost for the purposes of congestion control (the sender will half its congestion window). Detecting instances of 3-reordering is useful for determining the portion of reordered packets that are in fact as good as lost.

A sample's n-reordering may be expressed as a histogram, to summarize the frequency for each value of n.

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

A packet's n-reordering is sometimes equal to its reordering extent, e. n-reordering is different in the following ways:

1. n is a count of *adjacent* early packets.

2. Some reordered packets may not be n-reordered, but will have e (see the examples).

6. 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 [13]). In this case, the periodic sending interval should be chosen to reproduce the closest Src packet spacing expected. Of course, packet spacing is likely to vary as the stream traverses the test path.
<<<<Ed.Note: Is this sufficient? 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.

In practice, there may be limited ability to determine reordering extent, because the storage for previous packets may be limited. Saving only packets that indicate discontinuities (and their arrival positions) will reduce storage volume. When discarding all stream information beyond a threshold packet count, the reordering extent or degree of n-reordering may need to be expressed as greater than the threshold value, and Gap calculations would not be possible.

The requirement to ignore duplicate packets also requires storage. Here, tracking the sequence numbers of missing packets may minimize

storage. Missing packets may eventually be declared lost, or reordered if they arrive. The missing packet list and the largest sequence number received thus far are sufficient information to determine if a packet is a duplicate.

7. 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.
Throughout this section, we will refer to packets by their source sequence number, except where noted. So "Packet 4" refers to the packet with source sequence number 4, and the reader should refer to the tables in each example to determine packet 4's arrival index number, if needed.

Table 1 gives a simple case of reordering, where one packet is reordered, Packet 4. Packets are listed 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 reordering 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 j=7, 8 > 4, so the reordering extent is 1 or more.
when j=6, 7 > 4, so the reordering extent is 2 or more.
when j=5, 6 > 4, so the reordering extent is 3 or more.
when j=4, 5 > 4, so the reordering extent is 4 or more.
when j=3, but 3 < 4, and 4 is the maximum extent, e=4 (assuming there are no earlier sequence discontinuities, as in this example).

Further, we can compute the Late Time (210-148=62ms using DstTime) compared to Packet 5's arrival. If the receiver 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.

Following the definitions of section 5.1, Packet 4 is defined to be 4-reordered.

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

s		Src	Dst			Dst	Byte	Late
@Dst	NextExp	Time	Time	Delay	IPDV	Order	Offset	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 first:

when j=5, 7 > 5, so the reordering extent is 1 or more.
when j=4, but 4 < 5, so 1 is its maximum extent, and e=1.

Considering Packet 6 next:
when j=6, 5 < 6, the extent is not yet defined.
when j=5, 7 > 6, so the reordering extent is i-j=2 or more.
when j=4, 4 < 6, and we find 2 is its maximum extent, and e=2.

We can also associate each of these reordered packets with a reordering discontinuity. We find the minimum j=5 (for both packets) according to Section 4.2.4. So Packet 6 is associated with the same reordering discontinuity as Packet 5, at Packet 7.

Following the definitions of section 5.1, Packet 5 is defined to be 1-reordered, but Packet 6 is not qualified n-reordered.

A hypothetical sender/receiver pair may retransmit Packet 5 unnecessarily, since it is 1-reordered (in agreement with the singleton metric). Though Packet 6 may not be unnecessarily retransmitted, the receiver cannot advance Packet 7 to the higher layers until after Packet 6 arrives. Therefore, the singleton metric correctly determined that Packet 6 is reordered.

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		Src	Dst			Dst	Byte	Late
@Dst	NextExp	Time	Time	Delay	IPDV	Order	Offset	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	-88	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 (e) would be:

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

We first calculate the n-reordering. Considering Packet 4 first:
when n=1, 7<=j<8, and 10> 4, so the packet is 1-reordered.
when n=2, 6<=j<8, and 9 > 4, so the packet is 2-reordered.
when n=3, 5<=j<8, and 8 > 4, so the packet is 3-reordered.
when n=4, 4<=j<8, and 7 > 4, so the packet is 4-reordered.
when n=5, 3<=j<8, but 3 < 4, and 4 is the maximum n-reordering.

Considering packet 5[9] next:
when n=1, 8<=j<9, but 4 < 5, so the packet at i=9 is not qualified as n-reordered. We find the same to for Packet 6.

We now consider whether reordered Packets 5 and 6 are associated with the same reordering discontinuity as Packet 4. Using the test of Section 4.2.4, Definition 2, we find that the minimum j=4 for all three packets. They are all associated with the reordering discontinuity at Packet 7.

This example shows again that the n-reordering definition identifies a single Packet (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 5 and 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 associated with a single reordering discontinuity).

Table 4 Example with Packets Multiple Reordering Discontinuities
Sending order(s @Src): 1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16

```
       Discontinuity         Discontinuity
             |---------Gap---------|
s = 1, 2, 3, 6, 7, 4, 5, 8, 9, 10, 12, 13, 11, 14, 15, 16
i = 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16

r = 1, 2, 3, 4, 5, 1, 1, 2, 3,  4,  5,  6,  1,  2,  3,  4, ...
n =               1  2                    3
end r counts =    5  1                    6
```

Packet 4 has extent e=2, Packet 5 has extent e=3, and Packet 11 has e=2. There are two different reordering discontinuities, one at Packet 6 (where j=4) and one at Packet 12 (where j'=11).

According to the definition of Reordering Gap
Gap(j') = (j') - (j)
Gap(11) = (11) - (4) = 7

We also have three reordering-free runs of lengths 5, 1, and 6.

The differences between these two multiple-event metrics are evident here. Gaps are the distance between sequence discontinuities that are subsequently defined as reordering discontinuities, while reordering-free runs are capture the distance between reordered packets.

8. Security Considerations

8.1 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.

8.2 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.

8.3 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.

9. IANA Considerations

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

10. 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 Networking, 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 J.Bellardo and S.Savage, "Measuring Packet Reordering," Proceedings of the ACM SIGCOMM Internet Measurement Workshop 2002, November 6-8, Marseille, France.

10 S.Jaiswal et al., "Measurement and Classification of Out-of-Sequence Packets in a Tier-1 IP Backbone," Proceedings of the ACM SIGCOMM Internet Measurement Workshop 2002, November 6-8, Marseille, France.

11 L.Ciavattone, A.Morton, and G.Ramachandran, "Standardized Active Measurements on a Tier 1 IP Backbone," IEEE Communications Mag., pp 90-97, June 2003.

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

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

12. Acknowledgments

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

13. 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. Appendix on fragment order evaluation

Section 3 stated that fragment re-assembly is assumed prior to order evaluation, but that similar procedures could be applied prior to re-assembly. This appendix gives definitions and procedures to identify reordering in a packet stream that includes fragmentation.

The Metric retains the same name, Type-P-Reordered, but additional parameters are required.

This Appendix assumes that the device that divides a packet into fragments send them according to ascending fragment offset. Early Linux OS sent fragments in reverse order, so this possibility is worth checking.

13.1 Additional Metric Parameters:

+ MoreFrag, the state of the More Fragments Flag in the IP header

+ FragOffset, the offset from the beginning of a fragmented packet, in 8 octet units (also from the IP header).

+ FragSeq#, the sequence number from the IP header of a fragmented packet currently under evaluation for reordering. When set to zero, fragment evaluation is not in progress.

+ NextExpFrag, the Next Expected Fragment Offset at the Destination, in 8 octet units. Set to zero when fragment

evaluation is not in progress.

The packet sequence number, s, is assumed to be the same as the IP header sequence number. Also, the value of NextExp does not change with the in-order arrival of fragments. NextExp is only updated when a last fragment or a complete packet arrives.

Note that packets with missing fragments MUST be declared lost, and the Reordering status of any fragments that do arrive MUST be excluded from sample metrics.

13.2 Definition:

The value of Type-P-Reordered is typically false (the packet is in-order) when

* the sequence number s >= NextExp,

* AND the fragment offset FragOffset >= NextExpFrag

However, it more efficient to define reordered conditions exactly, and designate Type-P-Reordered as False otherwise.

The value of Type-P-Reordered is defined as True (the packet is reordered) under the conditions below. In these cases, the NextExp value does not change.

Case 1: if s < NextExp

Case 2: if s < FragSeq#

Case 3: if s>= NextExp AND s = FragSeq# AND FragOffset < NextExpFrag

This definition can also be illustrated in pseudo-code. A draft version of the code follows, and some simplification may be possible. A challenging aspect surrounds the housekeeping for the new parameters.

```
NextExp=0;
NextExpFrag=0;
FragSeq#=0;

while(packets arrive with s, MoreFrag, FragOffset)
{
if (s>=NextExp AND MoreFrag==0 AND s>=FragSeq#){
```



```
     /* a normal packet or last frag of an in-order packet arrived
*/
```

```
     NextExp = s+1;
     FragSeq# = 0;
     NextExpFrag = 0;
     Reordering = False;
     }
if (s>=NextExp AND MoreFrag==1 AND s>FragSeq#>=0){
     /* a fragment of a new packet arrived, possibly with a
     higher sequence number than the current fragmented packet */
     FragSeq# = s;
     NextExpFrag = FragOffset+1;
     Reordering = False;
     }
if (s>=NextExp AND MoreFrag==1 AND s==FragSeq#){
     /* a fragment of the "current packet s" arrived */
     if (FragOffset >= NextExpFrag){
             NextExpFrag = FragOffset+1;
             Reordering = False;
             }
     else{
             Reordering = True; /* fragment reordered  */
             }
     }
if (s>=NextExp AND MoreFrag==1 AND s < FragSeq#){
     /* case where a late fragment arrived */
     Reordering = True;
     }
else { /* when s < NextExp, or MoreFrag==0 AND s < FragSeq# */
     Reordering = True;
     }
}
```

A working version of the code would include a check to ensure that all fragments of a packet arrive before using the Reordered status further, such as in sample metrics.

13.3 Notes on Sample Metrics

All fragments with the same Source Sequence Number are assigned the same Source Time.

Evaluation with byte stream numbering may be simplified if the fragment offset is simply added to the SourceByte of the first packet (with fragment offset = 0), keeping the 8 octet units of the offset in mind.

14. Author's Addresses

Al Morton
AT&T Labs

Room D3 - 3C06
200 Laurel Ave. South
Middletown, NJ 07748 USA
Phone +1 732 420 1571
EMail: <acmorton@att.com>

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

Gomathi Ramachandran
AT&T Labs
Room C4 - 3D22
200 Laurel Ave. South
Middletown, NJ 07748 USA
Phone +1 732 420 2353
EMail: <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
Consultant
Calabasas, CA 91302 USA
Phone: + 1
EMail: <jerry@perser.org>

Full Copyright Statement

"Copyright (C) The Internet Society (date). All Rights Reserved.
This document and translations of it may be copied and furnished to
others, and derivative works that comment on or otherwise explain it
or assist in its implmentation may be prepared, copied, published
and distributed, in whole or in part, without restriction of any
kind, provided that the above copyright notice and this paragraph
are included on all such copies and derivative works. However, this
document itself may not be modified in any way, such as by removing
the copyright notice or references to the Internet Society or other
Internet organizations, except as needed for the purpose of

developing Internet standards in which case the procedures for copyrights defined in the Internet Standards process must be followed, or as required to translate it into languages other than English.

The limited permissions granted above are perpetual and will not be revoked by the Internet Society or its successors or assigns.

This document and the information contained herein is provided on an "AS IS" basis and THE INTERNET SOCIETY AND THE INTERNET ENGINEERING TASK FORCE DISCLAIMS ALL WARRANTIES, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO ANY WARRANTY THAT THE USE OF THE INFORMATION HEREIN WILL NOT INFRINGE ANY RIGHTS OR ANY IMPLIED WARRANTIES OF MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE.