

Network Working Group
Internet Draft
Document: <[draft-ietf-ippm-reordering-10.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

By submitting this Internet-Draft, each author represents that any applicable patent or other IPR claims of which he or she is aware have been or will be disclosed, and any of which he or she becomes aware will be disclosed, in accordance with [Section 6 of BCP 79](#).

This document is an Internet-Draft and is subject to all provisions of [section 3 of BCP 78](#).

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 obsoleted 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/lid-abstracts.txt>

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

Copyright Notice

Copyright (C) The Internet Society (2005).

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, including the context information required for all metrics. 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 for network characterization or receiver design. Additional metrics quantify the frequency of reordering and the

distance between separate occurrences. We then define a metric oriented toward reordering affects on TCP. Several examples of evaluation using the various sample metrics are included. An Appendix gives extended definitions for evaluating order with packet fragmentation.

Contents

Status of this Memo.....	1
Copyright Notice.....	1
Abstract.....	1
1 . Conventions used in this document.....	3
2 . Introduction.....	3
2.1 Motivation.....	4
2.2 Goals and Objectives.....	5
3 . A Reordered Packet Singleton Metric.....	6
3.1 Metric Name:.....	7
3.2 Metric Parameters:.....	7
3.3 Definition:.....	7
3.4 Sequence Discontinuity Definition.....	8
3.5 Evaluation of Reordering in Dimensions of Time or Bytes.....	9
3.6 Discussion.....	9
4 . Sample Metrics.....	10
4.1 Reordered Packet Ratio.....	10
4.1.1 Metric Name:.....	10
4.1.2 Metric Parameters:.....	10
4.1.3 Definition:.....	10
4.1.4 Discussion.....	11
4.2 Reordering Extent.....	11
4.2.1 Metric Name:.....	11
4.2.2 Parameter Notation:.....	11
4.2.3 Definition:.....	12
4.2.4 Discussion:.....	12
4.3 Reordering Late Time Offset.....	13
4.3.1 Metric Name:.....	13
4.3.2 Metric Parameters:.....	13
4.3.3 Definition:.....	13
4.3.4 Discussion.....	13
4.4 Reordering Byte Offset.....	14
4.4.1 Metric Name:.....	14
4.4.2 Metric Parameters:.....	14
4.4.3 Definition:.....	14
4.4.4 Discussion.....	15
4.5 Gaps between multiple Reordering Discontinuities.....	15

4.5.1	Metric Name:.....	15
4.5.2	Parameters:.....	15
4.5.3	Definition of Reordering Discontinuity:.....	16
4.5.4	Definition of Reordering Gap:.....	16
4.5.5	Discussion.....	16
4.6	Reordering-free Runs.....	17
4.6.1	Metric Name:.....	17
4.6.2	Parameters:.....	17

4.6.3	Definition:.....	17
4.6.4	Discussion:.....	18
5	Metrics Focused on Receiver Assessment: A TCP-Relevant Metric..	19
5.1	Metric Name:.....	19
5.2	Parameter Notation:.....	19
5.3	Definitions.....	19
5.4	Discussion:.....	20
6	Measurement and Implementation Issues.....	21
7	Examples of Arrival Order Evaluation.....	24
7.1	Example with a single packet reordered.....	24
7.2	Example with two packets reordered.....	25
7.3	Example with three packets reordered.....	27
7.4	Example with Multiple Packet Reordering Discontinuities.....	28
8	Security Considerations.....	28
8.1	Denial of Service Attacks.....	28
8.2	User data confidentiality.....	29
8.3	Interference with the metric.....	29
9	IANA Considerations.....	29
10	Normative References.....	29
11	Informative References.....	30
12	Acknowledgments.....	31
13	Appendix A Example Implementations in C (Informative).....	31
14	Appendix B Fragment Order Evaluation (Informative).....	34
14.1	Metric Name:.....	34
14.2	Additional Metric Parameters:.....	34
14.3	Definition:.....	34
14.4	Discussion: Notes on Sample Metrics when evaluating Fragments	36
15	Author's Addresses.....	36
	Full Copyright Statement.....	37
	Intellectual Property.....	37
	Acknowledgement.....	38

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](#) [[RFC2119](#)]. 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.

In this memo, the characters " \leq " should be read as "less than or equal to" and " \geq " as "greater than or equal to".

[2. Introduction](#)

Ordered arrival is a property found in packets that transit their path, where the packet sequence number increases with each new

Morton, et al. Standards Track exp. December 2005
Packet Reordering Metric for IPPM

Page 3
June 2005

arrival and there are no backward steps. The Internet Protocol [[RFC791](#)] has no mechanisms to assure either packet delivery or sequencing, and higher layer protocols (above IP) should be prepared to deal with both loss and reordering. This memo defines reordering metrics.

A unique sequence number, such as an incrementing message number 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](#) [[RFC2330](#)], 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 Paxson's reordering definition in [[Pax98](#)], 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 [RFC 793](#) [[RFC793](#)]. Earlier work to define ordered delivery includes [[Cia00](#)], [[Ben99](#)], [[Lou01](#)], [[Bel02](#)], [[Jai02](#)] and [[Cia03](#)].

[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 may change during transfer, and several specific path characteristics can make reordering more likely.

Examples are:

- * When two (or more) paths with slightly differing transfer times support a single packet stream or flow, then packets traversing the longer path(s) 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 (or parallel links) 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. The steady-state reordering condition typically causes an appreciable fraction of packets to be reordered. This form of reordering is most easily detected by minimizing the spacing between test packets. Transient reordering may occur in response to network instability; temporary routing loops can cause periods of extreme reordering. This condition is characterized by long in-order streams with occasional instances of reordering, sometimes with extreme correlation. However, we do not expect packet delivery in a completely random order, where for example, the last packet or the first packet in a sample is equally likely to arrive first at the destination. Thus we expect at least a minimal degree of order in the packet arrivals, as exhibited in real networks.

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 reordering has occurred.
2. Quantifying the degree of reordering. (We define a number of metrics to meet this goal, because receiving procedures differ by protocol or application. Since the affects of packet reordering vary with these procedures, a metric that quantifies a key aspect of one receiver's behavior could be irrelevant to a different receiver. If all the metrics defined below are reported, they give a wide-ranging view of reordering conditions.)

Reordering Metrics MUST:

- + have one or more applications, such as receiver design or network characterization, and a compelling relevance in the working group's view.
- + be computable "on the fly"
- + work even if the stream has duplicate or lost packets

It is desirable for Reordering Metrics to have one or more of the following attributes:

- + ability to concatenate results for segments measured separately to estimate the reordering of an entire path

Morton, et al. Standards Track exp. December 2005
Packet Reordering Metric for IPPM

Page 5
June 2005

- + simplicity for easy consumption and understanding
- + relevance to TCP design
- + relevance to Real-time application performance

The current set of metrics meet all the requirements above and provides all but the concatenation attribute (except in the case where segments exhibit no reordering, and one may estimate that the segment concatenation would also exhibit this desirable outcome). However, satisfying these goals restricts the set of metrics to those that provide some clear insight into network characterization or receiver design. They are not likely to be exhaustive in their coverage of reordering effects on applications, and additional measurements may be possible.

2.3 Required Context for All Reordering Metrics

A critical aspect of all reordering metrics is their inseparable bond with the measurement conditions. Packet reordering is not well defined unless the full measurement context is reported. Therefore, all reordering metric definitions include the following parameters:

1. The "Packet of Type-P" [[RFC2330](#)] identifiers for the packet stream, including the transport addresses for source and destination, and any other information which may result in different

packet treatments.

2. The stream parameter set for the sending discipline, such as the parameters unique to Periodic Streams (as in [RFC 3432](#) [[RFC3432](#)]), TCP-like Streams (as in [RFC 3148](#) [[RFC3148](#)]), or Poisson Streams (as in [RFC 2330](#) [[RFC2330](#)]). The stream parameters include the packet size, either specified as a fixed value or as a pattern of sizes (as applicable).

Whenever a metric is reported, it MUST include a description of these parameters to provide a context for the results.

[3.](#) A Reordered Packet Singleton Metric

The IPPM framework [RFC 2330](#) [[RFC2330](#)] 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 an algorithm (function) that produces a series of monotonically increasing identifiers applied to packets at the source to uniquely establish the order of packet transmission. The

unique sequence identifier may simply be an incrementing integer message number, as used below.

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 general, the receiver reproduces the sender's algorithm and the sequence of identifiers so that the "next expected" can be determined).

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 unique packet sequence number applied at the Source, in units of messages.
- + NextExp, the Next Expected Sequence number at the Destination, in units of messages. The stored value in NextExp is determined from a previously arriving packet.

And optionally:

- + PayloadSize, the number of bytes contained in the information field and referred to when the SrcByte sequence is based on bytes transferred.
- + SrcByte, the packet sequence number applied at the Source, in units of payload bytes.

3.3 Definition:

If a packet s , (sent at time, SrcTime) is found to be reordered by comparison with the Next Expected value, its Type-P-Reordered = TRUE; otherwise Type-P-Reordered = FALSE, as defined below:

The value of Type-P-Reordered is defined as TRUE if $s < \text{NextExp}$ (the packet is reordered). In this case, the NextExp value does not change.

The value of Type-P-Reordered is defined as FALSE if $s \geq \text{NextExp}$ (the packet is in-order). In this case, NextExp is set to $s+1$ for comparison with the next packet to arrive.

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 then /* s is in-order */
        NextExp = s + 1;
        Type-P-Reordered = False;
```



```
else      /* when s < NextExp */
          Type-P-Reordered = True
```

[3.4](#) Sequence Discontinuity Definition

Packets with $s > \text{NextExp}$ 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 [section 4](#)).

We define two different states for in-order packets.

When $s = \text{NextExp}$, the original sequence has been maintained, and there is no discontinuity present.

When $s > \text{NextExp}$, some packets in the original sequence have not yet arrived, and there is a sequence discontinuity associated with packet s . The size of the discontinuity is $s - \text{NextExp}$, equal to the number of packets presently missing, either reordered or lost.

In pseudo-code:

On successful arrival of a packet with sequence number s :

```
  if s >= NextExp, then /* s is in-order */
    if s > NextExp then
      SequenceDiscontinuity = True;
      SeqDiscontinuitySize = s - NextExp;
    else
      SequenceDiscontinuity = False;
      NextExp = s + 1;
      Type-P-Reordered = False;

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

Whether there are any Sequence Discontinuities and their size is determined by the conditions causing loss and/or reordering along the measurement path. Note that a packet could be reordered at one point, and subsequently lost elsewhere on the path, but this cannot be known from observations at the Destination.

[3.5](#) Evaluation of Reordering in Dimensions of Time or Bytes

It is possible to use alternate dimensions of time or payload bytes

to test for reordering in the definition of [section 3.3](#), as long as the SrcTimes and SrcBytes are unique and reliable. Sequence Discontinuities are easily defined and detected with message numbering, however, this is not so simple in the dimensions of time or bytes. This is a detractor for the alternate dimensions because the Sequence Discontinuity definition plays a key role in the sample metrics that follow.

It is possible to detect Sequence Discontinuities with payload byte numbering, but only when the complete pattern of payload sizes is stored at the Destination, or when payload size is constant and then the byte numbering adds needless complexity over message numbering.

It may be possible to detect Sequence Discontinuities with Periodic Streams and Source Time numbering, but there are practical pitfalls with sending exactly on-schedule and with clock reliability.

The dimensions of time and bytes remain an important basis for characterizing the extent of reordering, as described later.

[3.6](#) 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 (Type-P-Reordered=FALSE).

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](#). We provide a suggestion to minimize storage size needed in the section on Measurement and Implementation Issues.

[4](#). Sample Metrics

In this section, we define metrics applicable to a sample of packets from a single Source sequence number system. 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 metric highlights a relevant use.

The metrics in the sub-sections below have a network characterization orientation, but also have relevance to receiver design where reordering compensation is of interest. We begin with a simple ratio metric indicating the reordered portion of the sample.

[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 Streams (as in [RFC 2330](#) [[RFC2330](#)]), Periodic Streams (as in [RFC 3432](#) [[RFC3432](#)]), or TCP-like Streams (as in [RFC 3148](#) [[RFC3148](#)]), packet size or size patterns, and the following:

- + T₀, a start time
- + T_f, an end time
- + dT, a waiting time for each packet to arrive
- + K, the total number of packets in the stream sent from Source to Destination
- + L, the total number of packets received (arriving between T₀ and T_f+dT) out of the K packets sent. Recall that identical copies (duplicates) have been removed, so L ≤ K.

[4.1.3](#) Definition:

Given a stream of packets sent from a Source to a Destination, the ratio of reordered packets in the sample is

(Count of packets with Type-P-Reordered=TRUE) / (L)

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.

[4.1.4](#) Discussion

When the Type-P-Reordered-Ratio-Stream is zero, no further reordering metrics need be examined for that sample. Therefore, the value of this metric is its simple ability to summarize the results for a reordering-free sample.

[4.2](#) Reordering Extent

This section defines the extent to which packets are reordered, and associates a specific Sequence Discontinuity with each reordered packet. This section inherits the Parameters defined above.

[4.2.1](#) Metric Name:

Type-P-packet-Reordering-Extent-Stream

[4.2.2](#) Notation and Metric Parameters:

Recall that K is the number of packets in the stream at the Source and L is the number of packets received at the Destination.

Each packet has been assigned a sequence number, s , a consecutive integer from 1 to K in the order of packet transmission (at the source).

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

$s[i]$ can be thought of as a vector, where the index i is the arrival position of the packet with sequence number s . In theory, any Source sequence number could appear in any arrival position, but this is unlikely in reality.

Consider a reordered packet (Type-P-Reordered=TRUE) with arrival index i and source sequence number $s[i]$. There exists a set of indexes j ($1 \leq j < i$) such that $s[j] > s[i]$.

The new parameters are:

+ i , the index for arrival position, where $i-1$ represents an arrival earlier than i .

+ j , a set of one or more arrival indexes, where $1 \leq j < i$.

+ $s[i]$, the original sequence numbers, s , in order of arrival.

+ e , the Reordering Extent, defined below.

[4.2.3](#) Definition:

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

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 the 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. 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 (see the Byte Offset metric). With the bias understood, this definition is deemed sufficient, especially for those who demand "on the fly" calculations.

[4.2.4](#) Discussion:

The packet with index j ($s[j]$, identified in the Definition above) is the reordering discontinuity associated with packet s at 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.

If a receiver intends to restore order, then its buffer capacity determines its ability to handle 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 [section 4.4](#) on Reordering Byte Offset and the Examples section).

Although reordering extent primarily quantifies the offset in terms of arrival position, it may also be 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 Late Time Offset

Reordered packets can be assigned offset values indicating their lateness in terms of buffer time that a receiver must possess to accommodate them. Offset metrics are calculated only on reordered packets, as identified by the reordered packet singleton metric in [Section 3](#).

[4.3.1](#) Metric Name:

Type-P-packet-Late-Time-Stream

[4.3.2](#) Metric Parameters:

In addition to the parameters defined for Type-P-Reordered-Ratio-Stream, we specify:

- + DstTime, the time that each packet in the stream arrives at the destination, and may be associated with index i , or packet $s[i]$
- + LateTime($s[i]$), the offset of packet $s[i]$ in time, defined below

[4.3.3](#) Definition:

Lateness in time is calculated using destination times. When received packet $s[i]$ is reordered, and has a reordering extent e , then:

$$\text{LateTime}(s[i]) = \text{DstTime}(i) - \text{DstTime}(i - e)$$

Alternatively, using similar notation to that of [section 4.2](#), an equivalent definition is:

$$\text{LateTime}(s[i]) = \text{DstTime}(i) - \text{DstTime}(j), \text{ for } \min\{j \mid 1 \leq j < i\} \text{ that satisfies } s[j] > s[i].$$

[4.3.4](#) 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 time of storage prior to a cyclic play-out instant (as with de-jitter buffers).

Note that the One-way IPDV [[RFC3393](#)] 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 the destination has sufficient storage to accommodate the network's behavior and restore order. When an earlier packet in the Source sequence is lost, IPDV will necessarily be undefined for adjacent packets, and LateTime 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).

The combination of loss and reordering influences the LateTime metric. If presented with the arrival sequence 1, 10, 5 (where packets 2, 3, 4, and 6 through 9 are lost), LateTime would not indicate exactly how "late" packet 5 is from its intended arrival position. IPDV [[RFC3393](#)] would not capture this either, because of the lack of adjacent packet pairs. Assuming a Periodic Stream [[RFC3432](#)], an expected arrival time could be defined for all packets, but this is essentially a single-point delay variation metric (as defined in ITU-T Recommendations [[I.356](#)] and [[Y.1540](#)]), and not a reordering metric.

A sample's LateTime results may be expressed as a histogram, to summarize the frequency of buffer times needed to accommodate reordered packets and permit buffer tuning on that basis. A CDF with buffer time vs. percent of reordered packets accommodated may be informative.

[4.4](#) Reordering Byte Offset

Reordered packets can be assigned offset values indicating the storage in bytes that a receiver must possess to accommodate them. Offset metrics are calculated only on reordered packets, as identified by the reordered packet singleton metric in [Section 3](#).

[4.4.1](#) Metric Name:

Type-P-packet-Byte-Offset-Stream

[4.4.2](#) Metric Parameters:

We use the same parameters defined earlier, including the optional parameters of SrcByte and PayloadSize, and define:

+ ByteOffset(s[i]), the offset of packet s[i] in bytes

[4.4.3](#) Definition:

The Byte stream offset for reordered packet s[i] is the sum of the payload sizes of packets qualified by the following criteria:

- * Arrival prior to the reordered packet, s[i], and
- * The send sequence number, s, is greater than s[i].

Packets that meet both these criteria are normally buffered until the sequence beneath them is complete. Note that these criteria apply to both in-order and reordered packets.

For reordered packet s[i] with a reordering extent e:

$$\begin{aligned} \text{ByteOffset}(s[i]) &= \text{Sum}[\text{qualified packets}] \\ &= \text{Sum}[\text{PayloadSize}(\text{packet at } i-1 \text{ if qualified}), \\ &\quad \text{PayloadSize}(\text{packet at } i-2 \text{ if qualified}), \dots \\ &\quad \text{PayloadSize}(\text{packet at } i-e \text{ always qualified})] \end{aligned}$$

Using our earlier notation:

$$\text{ByteOffset}(s[i]) = \text{Sum}[\text{payloads of } s[j] \text{ where } s[j] > s[i] \text{ and } i > j \geq i-e]$$

[4.4.4](#) Discussion

We note that estimates of buffer size due to reordering depend on greatly on the test stream, in terms of the spacing between test packets and their size, especially when packet size is variable. In these and other circumstances, it may be most useful to characterize offset in terms of the payload size(s) of stored packets, using the Type-P-packet-Byte-Offset-Stream metric.

The byte offset metric can help predict whether reordered packets will be useful in a general receiver buffer system with finite limits. The limit is expressed as the number of bytes the buffer can store.

A sample's ByteOffset results may be expressed as a histogram, to

summarize the frequency of buffer lengths needed to accommodate reordered packets and permit buffer tuning on that basis. A CDF with buffer size vs. percent of reordered packets accommodated may be informative.

[4.5](#) Gaps between multiple Reordering Discontinuities

[4.5.1](#) Metric Name:

Type-P-packet-Reordering-Gap-Stream

[4.5.2](#) Parameters:

We use the same parameters defined earlier, but add the convention that index i' is greater than i , likewise $j' > j$, and define:

- + $\text{Gap}(s[j'])$, the Reordering Gap of packet $s[j']$ in units of integer messages
- + $\text{GapTime}(s[j'])$, the Reordering Gap of packet $s[j']$ in units of time

Morton, et al. Standards Track exp. December 2005
Packet Reordering Metric for IPPM

Page 15
June 2005

[4.5.3](#) Definition of Reordering Discontinuity:

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

Note that $s[j]$ will have been found to cause a sequence discontinuity, where $s > \text{NextExp}$ when evaluated with the reordered singleton metric as described in [section 3.4](#).

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

[4.5.4](#) 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:

$$\text{Gap}(s[j']) = (j') - (j)$$

Otherwise:

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

Gaps may also be expressed in time:

$$\text{GapTime}(s[j']) = \text{DstTime}(j') - \text{DstTime}(j)$$

[4.5.5 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. Gap lengths are also informative to receiver designers, revealing the period of reordering discontinuities. The combination of reordering gaps and extent reveals whether receivers will be required to handle cases of overlapping reordered packets.

[4.6 Reordering-free Runs](#)

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

[4.6.1 Metric Name:](#)

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

[4.6.2](#) Parameters:

We use the same parameters defined earlier, and define the following:

- + r, the run counter
- + x, the number of runs, also the number of reordered packets
- + a, the accumulator of in-order packets
- + p, the number of packets (when the stream is complete, $p=(x+a)=L$)
- + q, the sum of the squares of the runs counted

[4.6.3](#) Definition:

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

```
r = 0; /* r is the run counter */
x = 0; /* x 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 sum of the squares of the runs counted */
```

```
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 = 0;
        x++;
}
```

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 sum of the squared lengths.

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

The percent of packets in-order is $100 \cdot a/p$

The average Reordering-Free run length is a/x

The q counter gives an indication of variation of the Reordering-Free runs from the average by comparing q/a to a/x ($(q/a)/(a/x)$)

4.6.4 Discussion and Illustration:

Type-P-packet-Reordering-Free-Run-Stream parameters give a brief summary of the stream's reordering characteristics including the average reordering-free run length, and the variation of run lengths, therefore a key application of this metric is network evaluation.

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

$p = 36$
 $x = 3$
 $a = 33$
 $q = 3 * (11 \cdot 11) = 363$
ave reordering-free run = 11
 $q/a = 11$
 $(q/a)/(a/x) = 1.0$

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

$p = 36$
 $x = 3$
 $a = 33$
 $q = 1 + 1 + 961 = 963$
ave reordering-free run = 11
 $q/a = 29.18$
 $(q/a)/(a/x) = 2.65$

The variability in run length is prominent in the difference between the q values (sum of the squared run lengths) and comparing average run length to the $(q/a)/(a/x)$ ratios (equals 1 when all runs are the same length).

5. Metrics Focused on Receiver Assessment: A TCP-Relevant Metric

This section describes a metric that conveys information associated with the affect of reordering on TCP. However, in order to infer anything about TCP performance, the test stream MUST bear a close

resemblance to the TCP sender of interest. [RFC 3148](#) [[RFC3148](#)] lists the specific aspects of congestion control algorithms that must be specified. Further, [RFC 3148](#) recommends that Bulk Transfer Capacity metrics SHOULD have instruments to distinguish three cases of packet reordering (in [section 3.3](#)). The sample metrics defined above satisfy the requirements to classify packets that are slightly or grossly out-of-order. The metric in this section adds the capability to estimate whether reordering might cause the DUP-ACK threshold to be exceeded causing the Fast Retransmit algorithm to be invoked. Additional TCP Kernel Instruments are summarized in [[Mat03](#)].

[5.1](#) Metric Name:

Type-P-packet-n-Reordering-Stream

[5.2](#) 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 $s[1], s[2], \dots, s[l]$ be the original sequence numbers of the received packets, in the order of arrival.

[5.3](#) Definitions

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

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 packet 4 is n -reordered, with $n=3$.)

Definition 2: The degree of n -reordering of the sample is m/l , where m is the number of n -reordered packets in the sample.

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.

5.4 Discussion:

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

The n -reordering metric is helpful for matching the duplicate ACK threshold setting to a given path. For example, if a path exhibits no more than 5-reordering, a DUP-ACK threshold of 6 may avoid unnecessary retransmissions.

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, see [[RFC2581](#)]). 3 is default threshold for SCTP [[RFC2960](#)], and the future Datagram Congestion Control Protocol (DCCP).

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. Both time and position influence the sender's behavior.

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

1. n is a count of early packets with consecutive arrival positions at the receiver.
2. Reordered packets (Type-P-Reordered=TRUE) may not be n -reordered, but will have an extent, e (see the examples).

6. Measurement and Implementation Issues

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

In order to gauge the reordering for an application according to the metrics defined in this memo, it is RECOMMENDED to use the same sending pattern as the application of interest. In any case, the exact method of packet generation MUST be reported with the measurement results, including all stream parameters.

- + To make inferences about applications that use TCP, it is REQUIRED to use TCP-like Streams as in [[RFC3148](#)]
- + For real-time applications, it is RECOMMENDED to use Periodic Streams as in [[RFC3432](#)]

It is acceptable to report the metrics of Sections [3](#) and [4](#) with other IPPM metrics using Poisson Streams [[RFC2330](#)]. Poisson streams represent an "unbiased sample" of network performance for packet loss and delay metrics. However, it would be incorrect to make inferences about the application categories above using reordering metrics measured with Poisson streams.

Test stream designers may prefer to use a periodic sending interval so that a known temporal bias is maintained, also bringing simplified results analysis (as described in [[RFC3432](#)]). In this case, it is RECOMMENDED that the periodic sending interval should be chosen to reproduce the closest Source packet spacing expected. Testers must recognize that streams sent at the link speed serialization limit MUST have limited duration and MUST consider packet loss as an indication that the stream has caused congestion, and suspend further testing.

When intending to compare independent measurements of reordering, it is RECOMMENDED to use the same test stream parameters in each measurement system.

Packet lengths might also be varied to attempt to detect instances of parallel processing (they may cause steady state reordering). For example, a line-speed burst of the longest (MTU-length) packets

followed by a burst of the shortest possible packets may be an effective detecting pattern. Other size patterns are possible.

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 (message number) is included in the packet payload (possibly in application headers such as RTP), reordering metrics may be evaluated in a passive measurement arrangement. Also, it is possible to evaluate order at any point along a Source-Destination path, recognizing that intermediate measurements may differ from those made at the Destination (where the reordering affect on applications can be inferred).

It is possible to apply these metrics to evaluate reordering in a TCP sender's stream. In this case, the Source sequence numbers would be based on byte stream, or segment numbering. Since the stream may include retransmissions due to loss or reordering, care must be taken to avoid declaring retransmitted packets reordered. The additional sequence reference of `s` or `SrcTime` helps to avoid this ambiguity, or the optional TCP timestamp field [[RFC1323](#)].

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.

Ideally, the test instrument would have the ability to use all earlier packets at any point in the test stream. In practice there will be limited ability to determine reordering extent, due to the storage requirements for previous packets. Saving only packets that indicate discontinuities (and their arrival positions) will reduce storage volume.

Another solution is to use a sliding history window of packets, where the window size would be determined by an upper bound on the useful reordering extent. This bound could be several packets or several seconds-worth of packets, depending on the intended analysis. When discarding all stream information beyond the window, the reordering extent or degree of n-reordering may need to be expressed as greater than the window length if the reordering discontinuity information has been discarded, and Gap calculations would not be possible.

The requirement to ignore duplicate packets also mandates storage. Here, tracking the sequence numbers of missing packets may minimize storage size. Missing packets may eventually be declared lost, or reordered if they arrive. The missing packet list and the largest sequence number received thus far ($\text{NextExp} - 1$) are sufficient information to determine if a packet is a duplicate (assuming a manageable storage size for packets that are missing due to loss).

It is important to note that practical IP networks also have limited ability to "store" packets, even when routing loops appear temporarily. Therefore, the storage for reordering metrics (and their complexity) would only approach the number packets in the sample, K , when the sending time for K packets is small with respect to the network's largest possible transfer time.

Last, we note that determining reordering extents and gaps is tricky when there are overlapped or nested events. Test instrument complexity and reordering complexity are directly correlated.

7. Examples of Arrival Order Evaluation

This section provides some examples to illustrate how the non-reversing order criterion works, how n-reordering works in comparison, and the value of quantifying reordering in all the dimensions of time, bytes, 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.

7.1 Example with a single packet reordered

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. All packets contain PayloadSize=100 bytes, with SrcByte=(s x 100)-99 for s=1,2,3,4,...

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

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

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:

Morton, et al. Standards Track exp. December 2005
Packet Reordering Metric for IPPM

Page 24
June 2005

Using the notation $\langle s[1], \dots, s[i], \dots, s[L] \rangle$, the received packets are represented as:

$$\begin{array}{c} \backslash \\ s = 1, 2, 3, 5, 6, 7, 8, 4, 9, 10 \\ i = 1, 2, 3, 4, 5, 6, 7, 8, 9, 10 \\ / \end{array}$$

Applying the definition of Type-P-packet-Reordering-Extent-Stream:
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 ($210-148=62\text{ms}$ 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 indicate unusual behavior for Packet 4. Also, if Packet 4 had arrived at least 62ms earlier, it would have been in-order in this example.

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 designated 4-reordered.

[7.2](#) Example with two packets 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	Delay	IPDV	Dst Order	Byte Offset	Late Time
1	1	68	68		1		
2	2	88	68	0	2		
3	3	108	68	0	3		
4	4	128	68	0	4		
7	5	188	68	-22	5		
5	8	189	109	41	6	100	1
6	8	190	90	-19	7	100	2
8	8	208	68	0	8		
9	9	228	68	0	9		
10	10	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 $\langle s[1], \dots, s[i], \dots, s[l] \rangle$, the received packets are represented as:

$$\begin{array}{c}
 \backslash \quad \backslash \\
 s = 1, 2, 3, 4, 7, 5, 6, 8, 9, 10 \\
 i = 1, 2, 3, 4, 5, 6, 7, 8, 9, 10 \\
 / \quad /
 \end{array}$$

Considering Packet 5 first:

when $j=5$, $7 > 5$, so the reordering extent is 1 or more.

when $j=4$, we have $4 < 5$, 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 4.2.3](#). So Packet 6 is associated with the same reordering discontinuity as Packet 5, the Reordering Discontinuity at Packet 7.

This is a case where reordering extent e would over-estimate the packet storage required to restore order. Only one packet storage is

required (to hold Packet 7), but $e=2$ for Packet 6.

Following the definitions of [section 5](#), Packet 5 is designated 1-reordered, but Packet 6 is not designated 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.

[7.3](#) Example with three packets 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	@Dst	Next	Src	Dst	Delay	IPDV	Dst	Byte	Late
		Exp	Time	Time			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 $\langle s[1], \dots, s[i], \dots, s[l] \rangle$, 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 $n=1$, $7 \leq j < 8$, and $10 > 4$, so the packet is 1-reordered.
when $n=2$, $6 \leq j < 8$, and $9 > 4$, so the packet is 2-reordered.
when $n=3$, $5 \leq j < 8$, and $8 > 4$, so the packet is 3-reordered.
when $n=4$, $4 \leq j < 8$, and $7 > 4$, so the packet is 4-reordered.
when $n=5$, $3 \leq j < 8$, but $3 < 4$, and 4 is the maximum n-reordering.

Considering packet 5[9] next:

when $n=1$, $8 \leq j < 9$, but $4 < 5$, so the packet at $i=9$ is not designated 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 4.2.3](#), 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 n-reordering that might cause one unnecessary packet retransmission by the New Reno TCP sender (with DUP-ACK threshold=3 or 4). 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 Type-P-Reordered = TRUE for Packets 5 and 6, and they are all associated with a single reordering discontinuity).

[7.4](#) Example with Multiple Packet Reordering Discontinuities

Table 4 Example with Multiple Packet 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, 0, 0, 1, 2, 3, 4, 5, 0, 1, 2, 3, ...
number of runs,n = 1 2 3
end r counts = 5 0 5
(these values are computed after the packet arrives)

```

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

$\text{Gap}(s[j']) = (j') - (j)$
 $\text{Gap}(\text{Packet } 12) = (11) - (4) = 7$

We also have three reordering-free runs of lengths 5, 0, and 5.

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 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 (source) to another host (destination) through intervening networks. This method could be abused for denial of service attacks directed at destination and/or the intervening network(s).

Administrators of source, destination, 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. In most cases, a hashing function will produce a value suitable for payload comparisons.

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 the destination 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. Normative References

- [RFC791] Postel, J., "Internet Protocol", STD 5, [RFC 791](#), September 1981.
Obtain via: <http://www.rfc-editor.org/rfc/rfc791.txt>
- [RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", [RFC 2119](#), March 1997.
Obtain via: <http://www.rfc-editor.org/rfc/rfc2119.txt>
- [RFC2330] Paxson, V., Almes, G., Mahdavi, J., and Mathis, M., "Framework for IP Performance Metrics", [RFC 2330](#), May 1998.
Obtain via: <http://www.rfc-editor.org/rfc/rfc2330.txt>

- [RFC3148] Mathis, M. and Allman, M., "A Framework for Defining Empirical Bulk Transfer Capacity Metrics", [RFC 3148](#), July 2001.
Obtain via: <http://www.rfc-editor.org/rfc/rfc3148.txt>
- [RFC3432] Raisanen, V., Grotefeld, G., and Morton, A., "Network

performance measurement with periodic streams", [RFC 3432](#), November 2002.

11. Informative References

- [Bel02] J.Bellardo and S.Savage, "Measuring Packet Reordering," Proceedings of the ACM SIGCOMM Internet Measurement Workshop 2002, November 6-8, Marseille, France.
- [Ben99] 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.
- [Cia00] 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>
- [Cia03] L.Ciavattone, A.Morton, and G.Ramachandran, "Standardized Active Measurements on a Tier 1 IP Backbone," IEEE Communications Mag., pp 90-97, June 2003.
- [I.356] ITU-T Recommendation I.356, "B-ISDN ATM layer cell transfer performance", March 2000.
- [Jai02] 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.
- [Lou01] 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.
- [Mat03] M. Mathis, J Heffner and R Reddy, "Web100: Extended TCP Instrumentation for Research, Education and Diagnosis", ACM Computer Communications Review, Vol 33, Num 3, July 2003. <http://www.web100.org/docs/mathis03web100.pdf>
- [Pax98] 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>.

- [RFC793] Postel, J., "Transmission Control Protocol", STD 7, [RFC](#)

[793](#), September 1981.

Obtain via: <http://www.rfc-editor.org/rfc/rfc793.txt>

- [RFC1323] Jacobson, V., Braden, R., and Borman, D., "TCP Extensions for High Performance", [RFC 1323](#), May 1992.
- [RFC2581] Allman, M., Paxson, V., and Stevens, W., "TCP Congestion Control", [RFC 2581](#), April 1999.
- [RFC2960] Stewart, R., et al., "Stream Control Transmission Protocol", [RFC 2960](#), October 2000.
- [RFC3393] Demichelis, C., and Chimento, P., "IP Packet Delay Variation Metric for IP Performance Metrics (IPPM)", [RFC 3393](#), November 2002.
- [Y.1540] ITU-T Recommendation Y.1540, "Internet protocol data communication service - IP packet transfer and availability performance parameters", December 2002.

[12](#). Acknowledgments

The authors would like to acknowledge many helpful discussions with Matt Zekauskas, Jon Bennett (who authored the sections on Reordering-Free Runs), and Matt Mathis. We thank David Newman, Henk Uijterwaal, Mark Allman, Vern Paxson, and Phil Chimento for their reviews and suggestions, and Michal Przybylski for sharing implementation experiences with us on the `ippm-list`. Anura Jayasumana and Nischal Piratla brought in recent work-in-progress. We gratefully acknowledge the foundation laid by the authors of the IP performance Framework [[RFC2330](#)].

[13](#). [Appendix A](#) Example Implementations in C (Informative)

Two example c-code implementations of reordering definitions follow:

Example 1 n-reordering =====

```
#include <stdio.h>
```

```
#define MAXN 100
```

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

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

```
/*
```

```
 * 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[MAXN];        /* We have m[j-1] == number of
                           * j-reordered packets. */
    int    ring[MAXN];    /* 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 < MAXN; j++) m[j] = 0;
    for (;(s = read_sequence_number())!= EOF;l++,r=(r+1)%MAXN) {
        for (j=0; j<min(l, MAXN)&&ring[loop(r-j-1)];j++) m[j]++;
        ring[r] = s;
    }
    for (j = 0; j < MAXN && 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 < MAXN) printf("no %d-reordering\n", j+1);
    else printf("only up to %d-reordering is handled\n", MAXN);
    exit(0);
}

/* Example 2    singleton and n-reordering comparison =====
   Author:  Jerry Perser 7-2002 (mod by acm 12-2004)
   Compile: $ gcc -o jpboth file.c
   Usage:   $ jpboth 1 2 3 7 8 4 5 6 (pkt sequence given on cmdline)
   Note to cut/pasters: line 59 may need repair
*/

#include <stdio.h>
```

```

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

```

Morton, et al. Standards Track exp. December 2005
 Packet Reordering Metric for IPPM

Page 32
 June 2005

```

/* Global counters */
int receive_packets=0;      /* number of received */
int reorder_packets_Al=0;   /* num reordered pkts (singleton) */
int reorder_packets_Stas=0; /* num reordered pkts(n-reordering)*/

/* 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[MAXN]; /* 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) % MAXN;
    for (j=0; j<min(l, MAXN) && 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_Al += testorder1(packet); // singleton
        reorder_packets_Stas += testorder2(packet); //n-reord.
    }
    printf("Received packets = %d, Singleton Reordered = %d, n-
reordered = %d\n", receive_packets, reorder_packets_Al,
reorder_packets_Stas );
    exit(0);
}

```

Reference

ISO/IEC 9899:1999 (E), as amended by ISO/IEC 9899:1999/Cor.1:2001 (E). Also published as:

The C Standard: Incorporating Technical Corrigendum 1, British Standards Institute, ISBN: 0-470-84573-2, Hardcover, 558 pages, September 2003.

[14. Appendix B](#) Fragment Order Evaluation (Informative)

[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.

[14.1](#) Metric Name:

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.

[14.2](#) 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.

[14.3](#) Definition:

Morton, et al. Standards Track exp. December 2005
Packet Reordering Metric for IPPM

Page 34
June 2005

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

- * the sequence number $s \geq \text{NextExp}$,
- * AND the fragment offset $\text{FragOffset} \geq \text{NextExpFrag}$

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 $s < \text{NextExp}$

Case 2: if $s < \text{FragSeq\#}$

Case 3: if $s \geq \text{NextExp}$ AND $s = \text{FragSeq\#}$ AND $\text{FragOffset} < \text{NextExpFrag}$

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,
    for illustration only, redundant with else below */
    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.

[14.4](#) Discussion: Notes on Sample Metrics when evaluating Fragments

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.

15. 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 C2 - 3D02
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

Morton, et al. Standards Track exp. December 2005
Packet Reordering Metric for IPPM

Page 36
June 2005

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>

Copyright (C) The Internet Society (2005).

This document is subject to the rights, licenses and restrictions contained in [BCP 78](#), and except as set forth therein, the authors retain all their rights.

This document and the information contained herein are provided on an "AS IS" basis and THE CONTRIBUTOR, THE ORGANIZATION HE/SHE REPRESENTS OR IS SPONSORED BY (IF ANY), THE INTERNET SOCIETY AND THE INTERNET ENGINEERING TASK FORCE DISCLAIM 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.

Intellectual Property

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

Copies of IPR disclosures made to the IETF Secretariat and any assurances of licenses to be made available, or the result of an attempt made to obtain a general license or permission for the use of such proprietary rights by implementers or users of this

Morton, et al. Standards Track exp. December 2005
Packet Reordering Metric for IPPM

Page 37
June 2005

specification can be obtained from the IETF on-line IPR repository at <http://www.ietf.org/ipr>.

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

Acknowledgement

Funding for the RFC Editor function is currently provided by the Internet Society.

