IP Performance Metrics Working Group
Internet Draft
Document: <draft-ietf-ippm-reordering-01.txt>
Category: Standards Track

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

Packet Reordering Metric for IPPM

Status of this Memo

1. Abstract

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

2. 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.

3. 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 the sending time of each packet or an incrementing message number carried in each packet establishes the Source Sequence.

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

This metric is consistent with RFC 2330 [3], and classifies arriving packets with sequence numbers smaller than their predecessors as out-of-order, or reordered. For example, if arriving packets are numbered 1,2,4,5,3, then packet 3 is reordered. This is equivalent to Paxon's reordering definition in [4], where "late" packets were declared reordered. The alternative is to emphasize "premature" packets instead (4 and 5 in the example). 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] and more ???.

3.1 Motivation

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

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

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

* If for any reason, the packets in a buffer are not serviced in the order of their arrival, their order will change.
* If packets in a flow are assigned to multiple buffers (following evaluation of traffic characteristics, for example), and the buffers have different occupations and/or service rates, then order will likely change.

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

3.2 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

4. An Ordered Arrival Singleton Metric

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

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

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

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

4.1 Metric Name:

Type-P-Non-Reversing-Order

4.2 Metric Parameters:

+ Src, the IP address of a host

+ Dst, the IP address of a host

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

+ SrcNum, the packet sequence number applied at the Src, in units of messages or bytes.

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

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

4.3 Definition:

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

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

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

```
     if n >= NextExp, /* n is in-order */
             then
             NextExp = n + PayloadSize + 1;
     else            /* when n < NextExp */
             designate packet n as reordered;
```

When using message-based sequence numbering or Src time, PayloadSize=0.

4.4 Discussion

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

5. Sample Metrics

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

5.1 n-Reordering

[Note: This is the 10/2002 definition of n-Reordering. This definition focuses on TCP sender and receiver behavior, and in particular, New Reno TCP behavior when n=3.]

Metric Name: Type-P-packet-n-reordering-Poisson/Periodic-Stream

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

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

Note: This definition is illustrated by C code in Appendix A. It computes n-reordering for a particular value of n (when actually writing applications that would report the metric, one would probably report it for several values of n, such as 1, 2, 3, 4 -- and maybe a few more consecutive values).

Claim: If a packet is n-reordered and 0 < n' < n, then the packet is also n'-reordered.

Let m be the number of n-reordered packets in the sample.

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

Definition 3: The degree of reordering of the sample is its degree of 1-reordering.
<<<<Ed.Note - Def. 3 is no longer true using Definition 1. Blocks of reordered packets are not classified in/out-of order equivalently by singleton metric in section 4. See the examples in Table 2 and 3 in section 7. It appears that packets with 1-reordering and higher may be a subset of the reordered packets as designated by the singleton, and this is TBD.

<<<<Ed.Note - Need to add a short subsection to define the metrics on "proportion of reordered packets in the sample".

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

Discussion:

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

For a given sample, the number of n-reordered packets is the number of packets that would be considered as good as lost by a receiver that uses a buffer of n packets to correct reordering.

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 3-reordered packets and therefore consider 3-reordering a loss event for the purposes of congestion control (the sender will half its congestion window). 3-reordering is useful for determining the portion of reordered packets that are in fact as good as lost.

n-reordering is particularly useful for determining the portion of reordered packets which 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).

5.2 Reordering Offset

Any packet whose sequence number causes the Next Expected value to increment by more than the usual increment indicates a discontinuity in the sequence. From this point on, any packets with sequence number less than the Next Expected value can be assigned Offset values indicating their position (in packets or bytes) and lateness in terms of time of arrival with respect to a sequence discontinuity. The various Offset metrics are calculated only on reordered packets, as defined in section 4.

5.2.1 Metric Name: Type-P-packet-Position-Offset-Poisson/Periodic-Stream

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

+ DstOrder, numerical order in which each packet in the stream arrives at Dst

Definition: Reordered packets are associated with a specific sequence discontinuity by determining which earlier packet's sequence number skipped over them. We calculate all expressions of Offset with respect to that packet. Position Offset is calculated from a Dst Order number assigned to each packet on arrival:

Position Offset =
DstOrder(reordered packet)-DstOrder(packet at discontinuity)

Using the notation of Section 5.1, an equivalent definition is:
The Position Offset of Reordered Packet i is m = i-j, for
min{j|1<=j<i} that satisfies s[j]> s[i].

A sample's position offset may be expressed as a histogram, to easily summarize the extent and frequency of various offsets.

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

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

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

Definition: Lateness in time is calculated using Dst times.

Late Time =
DstTime(reordered packet)-DstTime(packet at discontinuity)

Using similar notation to that of Section 5.1, an equivalent definition is:
The Late Time of Reordered Packet i is t = DstTime[i]-DstTime[j],
for min{j|1<=j<i} that satisfies s[j]>s[i], or
SrcTime[j]>SrcTime[i].

5.2.3 Metric Name: Type-P-packet-Byte-Offset-Poisson/Periodic-Stream

Metric Parameters: We use the same parameters defined above.

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


```
When reordered packet has DstOrder=m
     Byte Offset = Sum[PayloadSize(packet, DstOrder=m-1),
                       PayloadSize(packet, DstOrder=m-2), ...
                       PayloadSize(packet at discontinuity)]
```

5.2.4 Discussion

The Offset metrics can predict whether reordered packets will be useful in a general, but limited receiver buffer system. 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 [8] 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).

For a sample of packets in a stream, results may be reported as a ratio of reordered packets to total packets sent by the source during the test. If separate reordering events can be distinguished, then an event count may also be reported (along with the event description, such as the number of reordered packets and their offsets). The distribution of various Offset metrics may also be reported and summarized as average, range, etc.

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 [Ref to npmps]. In this case, the periodic sending interval

should be chosen to reproduce the closest Src packet spacing expected.
<<<<Ed.Note: Need to expand this further, it is a very important consideration.

The Non-reversing order criterion remains 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 out-of-sequence. 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 out-of-sequence. 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. Out-of-order packets coincident with sequence numbers reaching end-of-range must also be detected for proper application of correction factor.

7. Examples of Order Evaluation

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

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

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

SrcNum @Dst	NextExp	Src Time	Dst Time	Delay	IPDV	Dst Order	Posit. 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	4	62
9	9	160	228	68	0	9		
10	10	180	248	68	0	10		

Each column gives the following information:

SrcNum   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.
Posit.Offset The Position Offset of an out-of-order packet.
LateTime The lateness of an out-of-order packet, ms.

We can see that when packet 4 arrives, NextExp=9, and it is declared reordered. Further, we can compute the Offset of packet 4 in terms of position (8-4=4 using DstOrder) and Late Time (210-148=62ms using DstTime) compared to packet 5's arrival. If Dst has a de-jitter buffer that holds more than 4 packets, or at least 62 ms storage, packet 4 may be useful. Note that 1-way delay and IPDV also indicate unusual behavior for packet 4.

If all packets contained 100 byte payloads, then Byte Offset is equal to 500 bytes.

In the notation of n-reordering, <s[1], ..., s[i], ..., s[l]> the received packets are represented as:

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

when n=1, 7<=J<8, and 8 > 4, so the packet at i=8 is 1-reordered.
when n=2, 6<=J<8, and 7 > 4, so the packet at i=8 is 2-reordered.
when n=3, 5<=J<8, and 6 > 4, so the packet at i=8 is 3-reordered.
when n=4, 4<=J<8, and 5 > 4, so the packet at i=8 is 4-reordered.
when n=5, 3<=J<8, but 3 < 4, no more reordering.

We note that the Position Offset is equal to the Max(n) with n-reordering.

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

SrcNum @Dst	NextExp	Src Time	Dst Time	Delay	IPDV	Dst Order	Posit. Offset	Late Time
1	1	0	68	68		1		
2	2	20	88	68	0	2		
3	3	40	108	68	0	3		
4	4	60	128	68	0	4		
7	5	120	188	68	-22	5		
5	8	80	189	109	41	6	1	1
6	8	100	190	90	-19	7	2	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 declared out-of-order. Their positional offsets (6-5=1 and 7-5=2, using DstOrder again) and Late times (189-188=1, 190-188=2) are small.

In the notation of n-reordering, the received packets are represented as:

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

Considering packet 5[6] first:
when n=1, 5<=J<6, and 7 > 5, so the packet at i=6 is 1-reordered.
when n=2, 4<=J<6, but 4 < 5, same for all earlier packets.

Considering packet 6[7] next:
when n=1, 6<=J<7, and 5 < 6, so the packet at I=7 is not n-reordered for any n, even though:

when N=2, 5<=J<7, and 7 > 6,
because n-reordering requires s[j]>s[i]

for all j such that i-n <= j < i (see Definition 1 in section 5.1).

A hypothetical sender/receiver pair may retransmit packet 5[8] unnecessarily, since it is 1-reordered (in agreement with the singleton metric). However, the receiver cannot advance packet 7[5] to the higher layers until after packet 6[7] arrives. Therefore, the singleton metric correctly determined that 6[7] is reordered, and the n-reordering metric indicates that the hypothetical receiver can deal with its arrival efficiently (no unnecessary retransmission).

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

SrcNum @Dst	NextExp	Src Time	Dst Time	Delay	IPDV	Dst Order	Posit. Offset	Late Time
1	1	0	68	68		1		
2	2	20	88	68	0	2		
3	3	40	108	68	0	3		
7	4	120	188	68	-68	4		
8	8	140	208	68	0	5		
9	9	160	228	68	0	6		
10	10	180	248	68	0	7		
4	11	60	250	190	122	8	4	62
5	11	80	252	172	-18	9	5	64
6	11	100	256	156	-16	10	6	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 SrcNum 4,5,and 6). Delay, Late time, and Position 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 Position Offsets would be:

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

In the notation of n-reordering, the received packets are represented as:

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

Considering packet 4[8] first:

when n=1, 7<=J<8, and 10> 4, so the packet at i=8 is 1-reordered.
when n=2, 6<=J<8, and 9 > 4, so the packet at i=8 is 2-reordered.
when n=3, 5<=J<8, and 8 > 4, so the packet at i=8 is 3-reordered.
when n=4, 4<=J<8, and 7 > 4, so the packet at i=8 is 4-reordered.
when n=5, 3<=J<8, but 3 < 4, same for all earlier packets.

Considering packet 5[9] next:

when n=1, 8<=J<9, and 4 < 5, so the packet at I=9 is not n-reordered

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

8. Security Considerations [mostly borrowed from npmps]

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 Neteworking, vol.7, no.6, pp.789-798, December 1999.

8 Demichelis, C., and Chimento, P., "IP Packet Delay Variation Metric for IPPM", work in progress.

11. Acknowledgments

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

12. Appendix A (informative)

Two example c-code implementations of reordering definitions follow:

Example 1 n-reordering ==

```
#include <stdio.h>

#define MAX_N   100

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



```
/*
 * Read new sequence number and return it.  Return a sentinel value
of EOF
 * (at least once) when there are no more sequence numbers.  In this
example,
 * the sequence numbers come from stdin; in an actual test, they
would come
 * from the network.
 */
int
read_sequence_number()
{
        int             res, rc;
        rc = scanf("%d\n", &res);
        if (rc == 1) return res;
        else return EOF;
}


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

```
        int             l = 0;          /* Number of sequence
numbers read. */
        int             s;              /* Last sequence number
read. */
        int             j;


        for (j = 0; j < MAX_N; j++) m[j] = 0;
        for (; (s = read_sequence_number()) != EOF; l++, r = (r+1) %
MAX_N) {
                for (j=0; j<min(l, MAX_N) && s<ring[loop(r-j-1)];
j++) m[j]++;
                ring[r] = s;
        }
        for (j = 0; j < MAX_N && m[j]; j++)
                printf("%d-reordering = %f%%\n", j+1, 100.0*m[j]/(l-
j-1));
        if (j == 0) printf("no reordering\n");
        else if (j < MAX_N) printf("no %d-reordering\n", j+1);
        else printf("only up to %d-reordering is handled\n", MAX_N);
        exit(0);
}
```


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

```
#include <stdio.h>

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

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

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

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

```
                iReturn = 1;
        }
        return iReturn;
}

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

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

int main(int argc, char *argv[])
{
        int i, packet;
```



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

13. Author's Addresses

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

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

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

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

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

Full Copyright Statement