Internet Draft
Expiration Date: September 2003

Stanislav Shalunov
Internet2
March 2003

Definition of IP Packet Reordering Metric
<draft-shalunov-reordering-definition-02.txt>

1. Status of this Memo

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

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/1id-abstracts.txt

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

This memo provides information for the Internet community. This memo does not specify an Internet standard of any kind. Distribution of this memo is unlimited.

2. Abstract

Various pieces of network testing equipment currently often report a characteristic that is referred to as a `degree (or percentage) of packet reordering.' The way this metric is computed is often undocumented and it differs between vendors. Having a useful numeric measure of the degree of packet reordering is important for applications such as TCP and VoIP on different ends of the spectrum. However, the metric that makes sense for one application may have no or little applicability to another. This document introduces a definition of reordering metric that is hoped to be applicable to a number of different applications by parametrizing the metric.

3. N-Reordering Metric Definition

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.

Claim 1: 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.

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

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.

Note: If n is taken to be the number of duplicate acknowledgments after which a TCP sender will retransmit a packet and halve its congestion window, n-reordering is useful for determining the portion of reordered packets that are in fact as good as lost.

4. Examples

This section is non-normative.

Several examples can be helpful. The reader is encouraged to compile the program in Appendix A and to try different inputs (on a POSIX system, one convenient way to supply test inputs to the program is to running a command similar to the following in the command shell: `echo 1 2 3 4 5 6 7 8 9 10 | fmt 1 | ./reord').

In the following, we first specify the sample and then describe the degrees of reordering and give some comments.

In this example, no reordering is observed:
    1 2 3 4 5 6 7 8 9 10
    no reordering

Gaps in sequence numbers (which might or might not indicate losses) are not counted as reordering:
    1 3 4 5 6 7 8 9 10
    no reordering

In fact, the reordering metric could be applied to any strictly monotically increasing progression of sequence numbers (consecutive numbering is just a convenience for easier writing of the definition):
    0 1000 2000 3000 4000 5000 6000 7000 8000 9000
    no reordering

A single transposition of adjacent packets:
    1 2 4 3 5 6 7 8 9 10
    1-reordering = 10.000000%
    no 2-reordering
(Notice that since there is no 2-reordering, there is also no 3-reordering and, in general, no n-reordering for any n>2, according to claim 1.)

Two transpositions in different parts of the sequence:
    1 2 4 3 5 6 8 7 9 10
    1-reordering = 20.000000%
    no 2-reordering

A single packet `delayed by two places':
    1 2 4 5 3 6 7 8 9 10
    1-reordering = 10.000000%
    2-reordering = 10.000000%
    no 3-reordering
(In the sequence above, packet with sequence number 3 is 2-reordered

-- and, of course, 1-reordered according to claim 1.)

A single packet `delayed by three places':

```
    1 2 4 5 6 3 7 8 9 10
    1-reordering = 10.000000%
    2-reordering = 10.000000%
    3-reordering = 10.000000%
    no 4-reordering
```

A single packet (with sequence number 3) is delayed by three places here. Note how if these were TCP packets, the receiver would have sent three duplicate ACKs on seeing packets with numbers 4, 5, and 6.

The definition is geared towards counting `late' packets (primarily because TCP tolerates early packets much better than late packets). In the following sequence, a packet is `three places early':

```
    1 2 3 4 8 5 6 7 9 10
    1-reordering = 10.000000%
    no 2-reordering
```

In other (informal) words, the reordering definition recovers quickly from seeing early packets.

Note that since definition 2 contains division by l (rather than l-n) it is never possible to have 100% degree of n-reordering for any n:

```
    10 9 8 7 6 5 4 3 2 1
    1-reordering = 90.000000%
    2-reordering = 80.000000%
    3-reordering = 70.000000%
    4-reordering = 60.000000%
    5-reordering = 50.000000%
    6-reordering = 40.000000%
    7-reordering = 30.000000%
    8-reordering = 20.000000%
    9-reordering = 10.000000%
    no 10-reordering
```

5. RFC 2330 Considerations

Within the framework of [RFC2330], the n-reordering metrics can only be interpreted in a meaningful fashion if, along with the metrics themselves and sample size, type of each packet and time when each packet was sent is reported.

6. Area of Applicability and Choice of Parameter Values

Different applications will require different parameter values to obtain a metric that will be relevant to them.

For example, for a (hypothetical) VoIP application that has no buffer to accomodate reordering, 0-reordering metric on its traffic is meaningful. Namely, the sum of loss and 0-reordering will be the percentage of packets that the application cannot play back.

For bulk TCP, 3-reordering (plus loss) of its traffic will be more meaningful (because of Fast Retransmit).

If the metrics were to be computed with simulated traffic so that behavior of real applications with their real traffic could be extrapolated, different types of packets and different send schedules would of course be required to come up with meaningful numbers (e.g., not implying that these are necessarily the best choices, it could be evenly spaced stream of small UDP packets for VoIP or bursts of back-to-back MTU-sized TCP packets for TCP).

7. Security Considerations

This document doesn't define any protocol. The metric definition per se is believed to have no security implications.

8. IANA Considerations

This document requires nothing from IANA.

9. Acknowledgments

I would like to thank Matt Mathis for a long and fruitful discussion of TCP behavior in the case of presence of packet reordering.

10. Appendix A

```
#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;          /* Counter of sequence numbers. */
        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);
        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);
}
```

11. References

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

12. Author's Address

Stanislav Shalunov <shalunov@internet2.edu>

See http://www.internet2.edu/~shalunov/ for current snail mail address and telephone number.

Expiration date: September 2003