Network Working Group
Request for Comments: 1624
Updates: 1141
Category: Informational

A. Rijsinghani, Editor
Digital Equipment Corporation

May 1994

Computation of the Internet Checksum via Incremental Update

Status of this Memo

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.

Abstract

This memo describes an updated technique for incremental computation of the standard Internet checksum. It updates the method described in RFC 1141.

Table of Contents

1. Introduction

Incremental checksum update is useful in speeding up several types of operations routinely performed on IP packets, such as TTL update, IP fragmentation, and source route update.

RFC 1071, on pages 4 and 5, describes a procedure to incrementally update the standard Internet checksum. The relevant discussion, though comprehensive, was not complete. Therefore, RFC 1141 was published to replace this description on Incremental Update. In particular, RFC 1141 provides a more detailed exposure to the procedure described in RFC 1071. However, it computes a result for certain cases that differs

from the one obtained from scratch (one's complement of one's complement sum of the original fields).

For the sake of completeness, this memo briefly highlights key points from RFCs 1071 and 1141. Based on these discussions, an updated procedure to incrementally compute the standard Internet checksum is developed and presented.

2. Notation and Equations

Given the following notation:

```
HC  - old checksum in header
C   - one's complement sum of old header
HC' - new checksum in header
C'  - one's complement sum of new header
m   - old value of a 16-bit field
m'  - new value of a 16-bit field
```

RFC 1071 states that C' is:

```
C' = C + (-m) + m'     --     [Eqn. 1]
   = C + (m' - m)
```

As RFC 1141 points out, the equation above is not useful for direct use in incremental updates since C and C' do not refer to the actual checksum stored in the header. In addition, it is pointed out that RFC 1071 did not specify that all arithmetic must be performed using one's complement arithmetic.

Finally, complementing the above equation to get the actual checksum, RFC 1141 presents the following:

```
HC' = ~(C + (-m) + m')
    = HC + (m - m')
    = HC + m + ~m'     --     [Eqn. 2]
```

3. Discussion

Although this equation appears to work, there are boundary conditions under which it produces a result which differs from the one obtained by checksum computation from scratch. This is due to the way zero is handled in one's complement arithmetic.

In one's complement, there are two representations of zero: the all zero and the all one bit values, often referred to as +0 and -0. One's complement addition of non-zero inputs can produce -0 as a result, but never +0. Since there is guaranteed to be at least one

non-zero field in the IP header, and the checksum field in the protocol header is the complement of the sum, the checksum field can never contain ~(+0), which is -0 (0xFFFF). It can, however, contain ~(-0), which is +0 (0x0000).

RFC 1141 yields an updated header checksum of -0 when it should be +0. This is because it assumed that one's complement has a distributive property, which does not hold when the result is 0 (see derivation of [Eqn. 2]).

The problem is avoided by not assuming this property. The correct equation is given below:

```
HC' = ~(C + (-m) + m')     --     [Eqn. 3]
    = ~(~HC + ~m + m')
```

4. Examples

Consider an IP packet header in which a 16-bit field m = 0x5555 changes to m' = 0x3285. Also, the one's complement sum of all other header octets is 0xCD7A.

Then the header checksum would be:

```
HC = ~(0xCD7A + 0x5555)
   = ~0x22D0
   =  0xDD2F
```

The new checksum via recomputation is:

```
HC' = ~(0xCD7A + 0x3285)
    = ~0xFFFF
    =  0x0000
```

Using [Eqn. 2], as specified in RFC 1141, the new checksum is computed as:

```
HC' = HC + m + ~m'
    =  0xDD2F + 0x5555 + ~0x3285
    =  0xFFFF
```

which does not match that computed from scratch, and moreover can never obtain for an IP header.

Applying [Eqn. 3] to the example above, we get the correct result:

```
HC' = ~(C + (-m) + m')
    = ~(0x22D0 + ~0x5555 + 0x3285)
    = ~0xFFFF
    =  0x0000
```

5. Checksum verification by end systems

If an end system verifies the checksum by including the checksum field itself in the one's complement sum and then comparing the result against -0, as recommended by RFC 1071, it does not matter if an intermediate system generated a -0 instead of +0 due to the RFC 1141 property described here. In the example above:

```
0xCD7A + 0x3285 + 0xFFFF = 0xFFFF
0xCD7A + 0x3285 + 0x0000 = 0xFFFF
```

However, implementations exist which verify the checksum by computing it and comparing against the header checksum field.

It is recommended that intermediate systems compute incremental checksum using the method described in this document, and end systems verify checksum as per the method described in RFC 1071.

The method in [Eqn. 3] is slightly more expensive than the one in RFC 1141. If this is a concern, the two additional instructions can be eliminated by subtracting complements with borrow [see Sec. 7]. This would result in the following equation:

```
HC' = HC - ~m - m'    --    [Eqn. 4]
```

In the example shown above,

```
HC' = HC - ~m - m'
    =  0xDD2F - ~0x5555 - 0x3285
    =  0x0000
```

6. Historical Note

A historical aside: the fact that standard one's complement arithmetic produces negative zero results is one of its main drawbacks; it makes for difficulty in interpretation. In the CDC 6000 series computers [4], this problem was avoided by using subtraction as the primitive in one's complement arithmetic (i.e., addition is subtraction of the complement).

7. Acknowledgments

The contribution of the following individuals to the work that led to this document is acknowledged:

Manu Kaycee – Ascom Timeplex, Incorporated
Paul Koning – Digital Equipment Corporation
Tracy Mallory – 3Com Corporation
Krishna Narayanaswamy – Digital Equipment Corporation
Atul Pandya – Digital Equipment Corporation

The failure condition was uncovered as a result of IP testing on a product which implemented the RFC 1141 algorithm. It was analyzed, and the updated algorithm devised. This algorithm was also verified using simulation. It was also shown that the failure condition disappears if the checksum verification is done as per RFC 1071.

8. Security Considerations

Security issues are not discussed in this memo.

9. Conclusions

It is recommended that either [Eqn. 3] or [Eqn. 4] be the implementation technique used for incremental update of the standard Internet checksum.

10. Author's Address

Anil Rijsinghani
Digital Equipment Corporation
550 King St
Littleton, MA 01460

Phone: (508) 486-6786
EMail: anil@levers.enet.dec.com

11. References

[1] Postel, J., "Internet Protocol - DARPA Internet Program Protocol Specification", STD 5, RFC 791, DARPA, September 1981.

[2] Braden, R., Borman, D., and C. Partridge, "Computing the Internet Checksum", RFC 1071, ISI, Cray Research, BBN Laboratories, September 1988.

[3] Mallory, T., and A. Kullberg, "Incremental Updating of the Internet Checksum", RFC 1141, BBN Communications, January 1990.

[4] Thornton, J., "Design of a Computer -- the Control Data 6600", Scott, Foresman and Company, 1970.