Network Working Group
Request for Comments: 1982
Updates: 1034, 1035
Category: Standards Track

R. Elz
University of Melbourne
R. Bush
RGnet, Inc.

August 1996

Serial Number Arithmetic

Status of this Memo

Abstract

This memo defines serial number arithmetic, as used in the Domain Name System. The DNS has long relied upon serial number arithmetic, a concept which has never really been defined, certainly not in an IETF document, though which has been widely understood. This memo supplies the missing definition. It is intended to update RFC1034 and RFC1035.

1. Introduction

The serial number field of the SOA resource record is defined in RFC1035 as

SERIAL The unsigned 32 bit version number of the original copy of the zone. Zone transfers preserve this value. This value wraps and should be compared using sequence space arithmetic.

RFC1034 uses the same terminology when defining secondary server zone consistency procedures.

Unfortunately the term "sequence space arithmetic" is not defined in either RFC1034 or RFC1035, nor do any of their references provide further information.

This phrase seems to have been intending to specify arithmetic as used in TCP sequence numbers [RFC793], and defined in [IEN-74].

Unfortunately, the arithmetic defined in [IEN-74] is not adequate for the purposes of the DNS, as no general comparison operator is

defined.

To avoid further problems with this simple field, this document defines the field and the operations available upon it. This definition is intended merely to clarify the intent of RFC1034 and RFC1035, and is believed to generally agree with current implementations. However, older, superseded, implementations are known to have treated the serial number as a simple unsigned integer, with no attempt to implement any kind of "sequence space arithmetic", however that may have been interpreted, and further, ignoring the requirement that the value wraps. Nothing can be done with these implementations, beyond extermination.

2. Serial Number Arithmetic

Serial numbers are formed from non-negative integers from a finite subset of the range of all integer values. The lowest integer in every subset used for this purpose is zero, the maximum is always one less than a power of two.

When considered as serial numbers however no value has any particular significance, there is no minimum or maximum serial number, every value has a successor and predecessor.

To define a serial number to be used in this way, the size of the serial number space must be given. This value, called "SERIAL_BITS", gives the power of two which results in one larger than the largest integer corresponding to a serial number value. This also specifies the number of bits required to hold every possible value of a serial number of the defined type. The operations permitted upon serial numbers are defined in the following section.

3. Operations upon the serial number

Only two operations are defined upon serial numbers, addition of a positive integer of limited range, and comparison with another serial number.

3.1. Addition

Serial numbers may be incremented by the addition of a positive integer n, where n is taken from the range of integers [0 .. (2^(SERIAL_BITS - 1) - 1)]. For a sequence number s, the result of such an addition, s', is defined as

$$s' = (s + n) \text{ modulo } (2 \hat{} \text{SERIAL\_BITS})$$

where the addition and modulus operations here act upon values that are non-negative values of unbounded size in the usual ways of integer arithmetic.

Addition of a value outside the range
[0 .. (2^(SERIAL_BITS - 1) - 1)] is undefined.

3.2. Comparison

Any two serial numbers, s1 and s2, may be compared. The definition of the result of this comparison is as follows.

For the purposes of this definition, consider two integers, i1 and i2, from the unbounded set of non-negative integers, such that i1 and s1 have the same numeric value, as do i2 and s2. Arithmetic and comparisons applied to i1 and i2 use ordinary unbounded integer arithmetic.

Then, s1 is said to be equal to s2 if and only if i1 is equal to i2, in all other cases, s1 is not equal to s2.

s1 is said to be less than s2 if, and only if, s1 is not equal to s2, and

```
(i1 < i2 and i2 - i1 < 2^(SERIAL_BITS - 1)) or
(i1 > i2 and i1 - i2 > 2^(SERIAL_BITS - 1))
```

s1 is said to be greater than s2 if, and only if, s1 is not equal to s2, and

```
(i1 < i2 and i2 - i1 > 2^(SERIAL_BITS - 1)) or
(i1 > i2 and i1 - i2 < 2^(SERIAL_BITS - 1))
```

Note that there are some pairs of values s1 and s2 for which s1 is not equal to s2, but for which s1 is neither greater than, nor less than, s2. An attempt to use these ordering operators on such pairs of values produces an undefined result.

The reason for this is that those pairs of values are such that any simple definition that were to define s1 to be less than s2 where (s1, s2) is such a pair, would also usually cause s2 to be less than s1, when the pair is (s2, s1). This would mean that the particular order selected for a test could cause the result to differ, leading to unpredictable implementations.

While it would be possible to define the test in such a way that the inequality would not have this surprising property, while being defined for all pairs of values, such a definition would be

unnecessarily burdensome to implement, and difficult to understand, and would still allow cases where

```
s1 < s2 and (s1 + 1) > (s2 + 1)
```

which is just as non-intuitive.

Thus the problem case is left undefined, implementations are free to return either result, or to flag an error, and users must take care not to depend on any particular outcome. Usually this will mean avoiding allowing those particular pairs of numbers to co-exist.

The relationships greater than or equal to, and less than or equal to, follow in the natural way from the above definitions.

4. Corollaries

These definitions give rise to some results of note.

4.1. Corollary 1

For any sequence number s and any integer n such that addition of n to s is well defined, (s + n) >= s. Further (s + n) == s only when n == 0, in all other defined cases, (s + n) > s.

4.2. Corollary 2

If s’ is the result of adding the non-zero integer n to the sequence number s, and m is another integer from the range defined as able to be added to a sequence number, and s" is the result of adding m to s’, then it is undefined whether s" is greater than, or less than s, though it is known that s" is not equal to s.

4.3. Corollary 3

If s" from the previous corollary is further incremented, then there is no longer any known relationship between the result and s.

4.4. Corollary 4

If in corollary 2 the value (n + m) is such that addition of the sum to sequence number s would produce a defined result, then corollary 1 applies, and s" is known to be greater than s.

5. Examples

5.1. A trivial example

The simplest meaningful serial number space has SERIAL_BITS == 2. In this space, the integers that make up the serial number space are 0, 1, 2, and 3. That is, 3 == 2^SERIAL_BITS - 1.

In this space, the largest integer that it is meaningful to add to a sequence number is 2^(SERIAL_BITS - 1) - 1, or 1.

Then, as defined 0+1 == 1, 1+1 == 2, 2+1 == 3, and 3+1 == 0. Further, 1 > 0, 2 > 1, 3 > 2, and 0 > 3. It is undefined whether 2 > 0 or 0 > 2, and whether 1 > 3 or 3 > 1.

5.2. A slightly larger example

Consider the case where SERIAL_BITS == 8. In this space the integers that make up the serial number space are 0, 1, 2, ... 254, 255. 255 == 2^SERIAL_BITS - 1.

In this space, the largest integer that it is meaningful to add to a sequence number is 2^(SERIAL_BITS - 1) - 1, or 127.

Addition is as expected in this space, for example: 255+1 == 0, 100+100 == 200, and 200+100 == 44.

Comparison is more interesting, 1 > 0, 44 > 0, 100 > 0, 100 > 44, 200 > 100, 255 > 200, 0 > 255, 100 > 255, 0 > 200, and 44 > 200.

Note that 100+100 > 100, but that (100+100)+100 < 100. Incrementing a serial number can cause it to become "smaller". Of course, incrementing by a smaller number will allow many more increments to be made before this occurs. However this is always something to be aware of, it can cause surprising errors, or be useful as it is the only defined way to actually cause a serial number to decrease.

The pairs of values 0 and 128, 1 and 129, 2 and 130, etc, to 127 and 255 are not equal, but in each pair, neither number is defined as being greater than, or less than, the other.

It could be defined (arbitrarily) that 128 > 0, 129 > 1, 130 > 2, ..., 255 > 127, by changing the comparison operator definitions, as mentioned above. However note that that would cause 255 > 127, while (255 + 1) < (127 + 1), as 0 < 128. Such a definition, apart from being arbitrary, would also be more costly to implement.

6. Citation

As this defined arithmetic may be useful for purposes other than for the DNS serial number, it may be referenced as Serial Number Arithmetic from RFC1982. Any such reference shall be taken as implying that the rules of sections 2 to 5 of this document apply to the stated values.

7. The DNS SOA serial number

The serial number in the DNS SOA Resource Record is a Serial Number as defined above, with SERIAL_BITS being 32. That is, the serial number is a non negative integer with values taken from the range [0 .. 4294967295]. That is, a 32 bit unsigned integer.

The maximum defined increment is 2147483647 (2^31 - 1).

Care should be taken that the serial number not be incremented, in one or more steps, by more than this maximum within the period given by the value of SOA.expire. Doing so may leave some secondary servers with out of date copies of the zone, but with a serial number "greater" than that of the primary server. Of course, special circumstances may require this rule be set aside, for example, when the serial number needs to be set lower for some reason. If this must be done, then take special care to verify that ALL servers have correctly succeeded in following the primary server's serial number changes, at each step.

Note that each, and every, increment to the serial number must be treated as the start of a new sequence of increments for this purpose, as well as being the continuation of all previous sequences started within the period specified by SOA.expire.

Caution should also be exercised before causing the serial number to be set to the value zero. While this value is not in any way special in serial number arithmetic, or to the DNS SOA serial number, many DNS implementations have incorrectly treated zero as a special case, with special properties, and unusual behaviour may be expected if zero is used as a DNS SOA serial number.

8. Document Updates

RFC1034 and RFC1035 are to be treated as if the references to "sequence space arithmetic" therein are replaced by references to serial number arithmetic, as defined in this document.

9. Security Considerations

This document does not consider security.

It is not believed that anything in this document adds to any security issues that may exist with the DNS, nor does it do anything to lessen them.

References

[RFC1034] Domain Names - Concepts and Facilities,
P. Mockapetris, STD 13, ISI, November 1987.

[RFC1035] Domain Names - Implementation and Specification
P. Mockapetris, STD 13, ISI, November 1987

[RFC793] Transmission Control protocol
Information Sciences Institute, STD 7, USC, September 1981

[IEN-74] Sequence Number Arithmetic
William W. Plummer, BB&N Inc, September 1978

Acknowledgements

Thanks to Rob Austein for suggesting clarification of the undefined comparison operators, and to Michael Patton for attempting to locate another reference for this procedure. Thanks also to members of the IETF DNSIND working group of 1995-6, in particular, Paul Mockapetris.

Authors' Addresses

Robert Elz
Computer Science
University of Melbourne
Parkville, Vic, 3052
Australia.

EMail: kre@munnari.OZ.AU

Randy Bush
RGnet, Inc.
10361 NE Sasquatch Lane
Bainbridge Island, Washington, 98110
United States.

EMail: randy@psg.com