April Fools' Day RFCs

Adam Lamers

March 2023

Contents

1 Request for Comments

A Request for Comments (RFC), in the context of Internet governance, is a type of publication from the Internet Engineering Task Force (IETF) and the Internet Society (ISOC), usually describing methods, behaviors, research, or innovations applicable to the working of the Internet and Internet-connected systems [1]

2 April Fools' Day RFC

2.1 The Twelve Networking Truths

2.1.1 Introduction

This Request for Comments (RFC) provides information about the fundamental truths underlying all networking. These truths apply to networking in general, and are not limited to TCP/IP, the Internet, or any other subset of the networking community.

2.1.2 The Fundamental Truths

1. It Has To Work.
2. No matter how hard you push and no matter what the priority, you can't increase the speed of light.
3. (corollary). No matter how hard you try, you can't make a baby in much less than 9 months. Trying to speed this up **might** make it slower, but it won't make it happen any quicker.
4. With sufficient thrust, pigs fly just fine. However, this is not necessarily a good idea. It is hard to be sure where they are going to land, and it could be dangerous sitting under them as they fly overhead.
5. Some things in life can never be fully appreciated nor understood unless experienced firsthand. Some things in networking can never be fully understood by someone who neither builds commercial networking equipment nor runs an operational network.
6. It is always possible to aglutenate multiple separate problems into a single complex interdependent solution. In most cases this is a bad idea.
7. It is easier to move a problem around (for example, by moving the problem to a different part of the overall network architecture) than it is to solve it.
8. (corollary). It is always possible to add another level of indirection.
9. It is always something
10. (corollary). Good, Fast, Cheap: Pick any two (you can't have all three).
11. It is more complicated than you think.
12. For all resources, whatever it is, you need more.
13. (corollary) Every networking problem always takes longer to solve than it seems like it should.
14. One size never fits all.
15. Every old idea will be proposed again with a different name and a different presentation, regardless of whether it works.
16. In protocol design, perfection has been reached not when there is nothing left to add, but when there is nothing left to take away. [2]

2.2 The Address is the Message

Declaring that the **address** is the **message**, the IPng WG has selected a packet format which includes 1696 bytes of address space. This length is a multiple of 53 and is completely compatible with ATM architecture. Observing that it's not what you know but who you know, the IPng focused on choosing an addressing scheme that makes it possible to talk to everyone while dispensing with the irrelevant overhead of actually having to say anything.

Security experts hailed this as a major breakthrough. With no content left in the packets, all questions of confidentiality and integrity are moot. Intelligence and law enforcement agencies immediately refocused their efforts to detect who's talking to whom, and are silently thankful they can avoid divisive public debate about key escrow, export control and related matters.

Although the IPng WG declared there should be more than enough address space for everyone, service providers immediately began vying for reserved portions of the address space. [3]

2.3 Hyper Text Coffee Pot Control Protocol (HTCPCP/1.0)

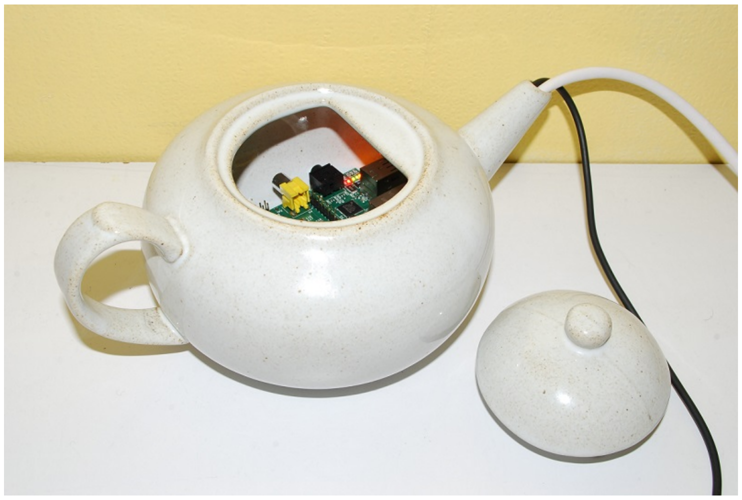

Figure 1: Back-end infrastructure of error418.net, which implements HTCPCP

There is ***coffee*** all over the world. Increasingly, in a world in which computing is ubiquitous, the computists want to make ***coffee***.***Coffee*** brewing is an art, but the distributed intelligence of the web- connected world transcends art. Thus, there is a strong, dark, rich requirement for a protocol designed espressoly for the brewing of ***coffee***. ***Coffee*** is brewed using ***coffee*** pots. Networked ***coffee*** pots require a control protocol if they are to be controlled.

Increasingly, home and consumer devices are being connected to the Internet. Early networking experiments demonstrated vending devices connected to the Internet for status monitoring [COKE]. One of the first remotely operated machine to be hooked up to the Internet, the Internet Toaster, (controlled via SNMP) was debuted in 1990 [RFC2235].

The demand for ubiquitous appliance connectivity that is causing the consumption of the IPv4 address space. Consumers want remote control of devices such as coffee pots so that they may wake up to freshly brewed ***coffee***, or cause coffee to be prepared at a precise time after the completion of dinner preparations. [4]

2.4 UTF-9 and UTF-18 Efficient Transformation Formats of Unicode

- UTF-9 (Unicode Transformation Format-9) is a variable-length character encoding format. The highest bit of each nonet (i.e. 9-bit byte) is used as the continuation flag, the remaining octet is the ISO-10646 character code. For characters 0x0-0xFF it means full backward compatibility. All Unicode characters can be encoded in UTF-9, so there is no need to use U+D800 - U+DBFF surrogate codes

- UTF-18 (Unicode Transformation Format-18) is a fixed-length character encoding format. It uses two nonets. Characters in the range U+0000 - U+2FFFF remain unchanged, and in the range U+E0000 - U+EFFFFF are shifted by 0x70000, to the range 0x30000 - 0x3ffff. Other values cannot be represented in UTF-18. [5]

UNICODE	UTF-9	UTF-18	UTF-8
U+0041	101	000101	101
U+0104	401 004	000404	304-204
U+AC00	654 000	126000	352 260 200

2.5 A Compact Representation of IPv6 Addresses

1. Abstract IPv6 addresses, being 128 bits long, need 32 characters to write in the general case, if standard hex representation, is used, plus more for any punctuation inserted (typically about another 7 characters, or 39 characters total). This document specifies a more compact representation of IPv6 addresses, which permits encoding in a mere 20 bytes.

2. Introduction

   It is always necessary to be able to write in characters the form of an address, though in actual use it is always carried in binary. For IP version 4 (IP Classic) the well known dotted quad format is used. That is, 10.1.0.23 is one such address. Each decimal integer represents a one octet of the 4 octet address, and consequently has a value between 0 and 255 (inclusive). The written length of the address varies between 7 and 15 bytes.

For IPv6 however, addresses are 16 octets long [IPv6], if the old standard form were to be used, addresses would be anywhere between 31 and 63 bytes, which is, of course, untenable.

Because of that, IPv6 had chosen to represent addresses using hex digits, and use only half as many punctuation characters, which will mean addresses of between 15 and 39 bytes, which is still quite long. Further, in an attempt to save more bytes, a special format was invented, in which a single run of zero octets can be dropped, the two adjacent punctuation characters indicate this has happened, the number of missing zeroes can be deduced from the fixed size of the address.

In most cases, using genuine IPv6 addresses, one may expect the address as written to tend toward the upper limit of 39 octets, as long strings of zeroes are likely to be rare, and most of the other groups of 4 hex digits are likely to be longer than a single non-zero digit (just as MAC addresses typically have digits spread throughout their length).

This document specifies a new encoding, which can always represent any IPv6 address in 20 octets. While longer than the shortest possible representation of an IPv6 address, this is barely longer than half the longest representation, and will typically be shorter than the representation of most IPv6 addresses.

3. Current formats

[AddrSpec] specifies that the preferred text representation of IPv6 addresses is in one of three conventional forms.

The preferred form is x:x:x:x:x:x:x:x, where the 'x's are the hexadecimal values of the eight 16-bit pieces of the address.

Examples:

FEDC:BA98:7654:3210:FEDC:BA98:7654:3210 (39 characters)

1080:0:0:0:8:800:200C:417A (25 characters)

The second, or zero suppressed, form allows "::" to indicate multiple groups of suppressed zeroes, hence:

1080:0:0:0:8:800:200C:417A

may be represented as

1080::8:800:200C:417A

a saving of just 5 characters from this typical address form, and still leaving 21 characters.

In other cases the saving is more dramatic, in the extreme case, the address:

0:0:0:0:0:0:0:0

that is, the unspecified address, can be written as

::

This is just 2 characters, which is a considerable saving. However such cases will rarely be encountered.

The third possible form mixes the new IPv6 form with the old IPv4 form, and is intended mostly for transition, when IPv4 addresses are embedded into IPv6 addresses. These can be considerably longer than the longest normal IPv6 representation, and will eventually be phased out. Consequently they will not be considered further here.

4. The New Encoding Format

   The new standard way of writing IPv6 addresses is to treat them as a 128 bit integer, encode that in base 85 notation, then encode that using 85 ASCII characters.

5. Why 85?

   2^{128} is 340282366920938463463374607431768211456. 85^{20} is 387595310845143558731231784820556640625, and thus in 20 digits of base 85 representation all possible 2^{128} IPv6 addresses can clearly be encoded.

   84^{20} is 305904398238499908683087849324518834176, clearly not sufficient, 21 characters would be needed to encode using base 84, this wastage of notational space cannot be tolerated.

   On the other hand,

$$94^{19} = 30862366077815087592879016454695419904$$

   , also insufficient to encode all 2^{128} different IPv6 addresses, so 20 characters would be needed even with base 94 encoding. As there are just 94 ASCII characters (excluding control characters, space, and del) base 94 is the largest reasonable value that can be used. Even if space were allowed, base 95 would still require 20 characters - equation 1.

   Thus, any value between 85 and 94 inclusive could reasonably be chosen. Selecting 85 allows the use of the smallest possible subset of the ASCII characters, enabling more characters to be retained for other uses, eg, to delimit the address.

6. The Character Set

   The character set to encode the 85 base85 digits, is defined to be, in ascending order:

   '0'..'9', 'A'..'Z', 'a'..'z', '!', '', '','')',' *',' +',' −',' ;',' <',' =',' >',' ?',' @',' ',' ','','`','','|','',and' '.

This set has been chosen with considerable care. From the 94 printable ASCII characters, the following nine were omitted:

"' and """, which allow the representation of IPv6 addresses to be quoted in other environments where some of the characters in the chosen character set may, unquoted, have other meanings.

',' to allow lists of IPv6 addresses to conveniently be written, and '.' to allow an IPv6 address to end a sentence without requiring it to be quoted.

'/' so IPv6 addresses can be written in standard CIDR address/length notation, and ':' because that causes problems when used in mail headers and URLs.

'[' and ']', so those can be used to delimit IPv6 addresses when represented as text strings, as they often are for IPv4,

And last, '¿ because it is often difficult to represent in a way where it does not appear to be a quote character, including in the source of this document.

7. Converting an IPv6 address to base 85.

   The conversion process is a simple one of division, taking the remainders at each step, and dividing the quotient again, then reading up the page, as is done for any other base conversion.

   For example, consider the address shown above

   1080:0:0:0:8:800:200C:417A

   In decimal, considered as a 128 bit number, that is 21932261930451111902915077091070067066.

   As we divide that successively by 85 the following remainders emerge: 51, 34, 65, 57, 58, 0, 75, 53, 37, 4, 19, 61, 31, 63, 12, 66, 46, 70, 68, 4.

   Thus in base85 the address is:

   4-68-70-46-66-12-63-31-61-19-4-37-53-75-0-58-57-65-34-51.

   Then, when encoded as specified above, this becomes:

   4)+kCVzJ4br¿0wv

   This procedure is trivially reversed to produce the binary form of the address from textually encoded format.

8. Additional Benefit

   Apart from generally reducing the length of an IPv6 address when encode in a textual format, this scheme also has the benefit of returning IPv6 addresses to a fixed length representation, leading zeroes are never omitted, thus removing the ugly and awkward variable length representation that has previously been recommended.

9. Implementation Issues

   Many current processors do not find 128 bit integer arithmetic, as required for this technique, a trivial operation. This is not considered a serious drawback in the representation, but a flaw of the processor designs.

   It may be expected that future processors will address this defect, quite possibly before any significant IPv6 deployment has been accomplished.

10. Security Considerations

    By encoding addresses in this form, it is less likely that a casual observer will be able to immediately detect the binary form of the address, and thus will find it harder to make immediate use of the address. As IPv6 addresses are not intended to be learned by humans, one reason for which being that they are expected to alter in comparatively short timespan, by human perception, the somewhat challenging nature of the addresses is seen as a feature.

Further, the appearance of the address, as if it may be random gibberish in a compressed file, makes it much harder to detect by a packet sniffer programmed to look for bypassing addresses. [6]

References

[1] Stephen D. Crocker, How the Internet Got Its Rules, The New York Times, 6 April 2009". The New York Times. April 7, 2009. Retrieved April 3, 2012.

[2] Network Working Group R. Callon, Editor Request for Comments: 1925 IOOF Category: Informational 1 April 1996

[3] S. Crocker Request for Comments: 1776 CyberCash, Inc. Category: Informational 1 April 1995

[4] Network Working Group L. Masinter Request for Comments: 2324 1 April 1998

[5] Network Working Group M. Crispin Request for Comments: 4042 Panda Programming Category: Informational 1 April 2005

[6] Network Working Group R. Elz Request for Comments: 1924 University of Melbourne Category: Informational 1 April 1996