SAND2016-4063C

Exceptional service in the national interest

Sandia National Laboratories

IPv6 Addresses

Big Numbers, Big Solutions

Andrew Steele
Andrew.Steele@sandia.gov
@ahsteele

Robert H. Engelhardt
Robert.Engelhardt@sanida.gov
@rheone

U.S. DEPARTMENT OF ENERGY

NNSA National Nuclear Security Administration

Sandia National Laboratories is a multi-program laboratory managed and operated by Sandia Corporation, a wholly owned subsidiary of Lockheed Martin Corporation, for the U.S. Department of Energy's National Nuclear Security Administration under contract DE-AC04-94AL85000. SAND NO. 2011-XXXXP

IPv6 Addresses: Big Numbers, Big Solutions

Sandia National Laboratories

Abstract numbers much bigger than 7 are hard to conceptualize. IPv6 and its 2^{128} possibilities are big, but developing with it doesn't have to be a big problem. Let us help you get on track with IPv6 addressing.

Our team at Sandia National Laboratories has come up with some techniques to model, manage, and manipulate those all-too-large 128-bit numbers that aren't typically accounted for in common 32-bit software environments. We will cover some of our exploration into the concepts needed, as well as the methods that we have developed for performing network mathematics in high level languages such as C# and SQL.

Since June 2015 we have been utilizing these techniques as the backbone for an application that manages the Sandia National Laboratories IPv4 and IPv6 networks, subnets and addresses.

- Big Numbers
- An Intro to IP Addresses
- Basics of Subnetting Math
- Programmatic Mathematics
- RDBMS Implementation

Sandia National Laboratories

BIG NUMBERS

like really big

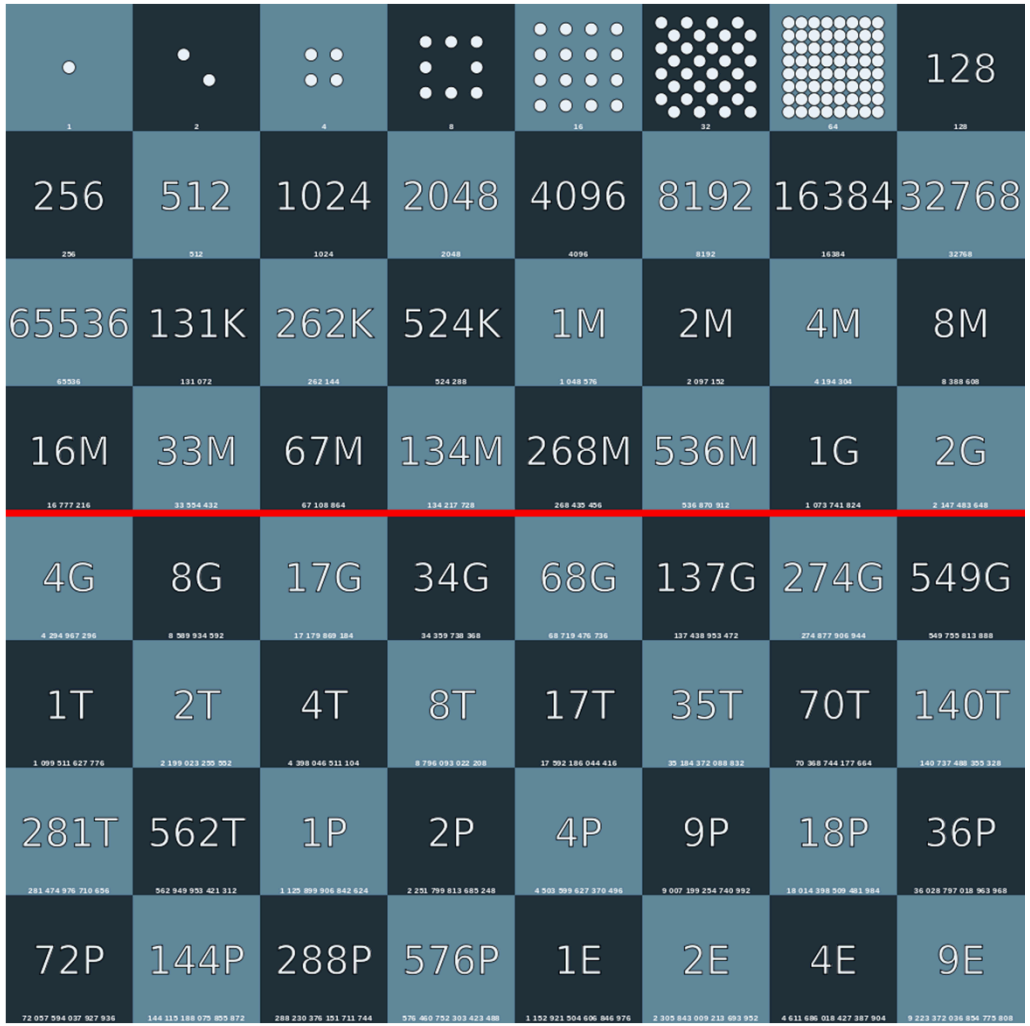
A Simple chess board

Sandia National Laboratories

If a chessboard were to have wheat placed upon each square such that one grain were placed on the first square, two on the second, four on the third, and so on (doubling the number of grains on each subsequent square), how many grains of wheat would be on the chessboard at the finish?

-Wikipedia “Wheat and chessboard problem”

A Simple chess board



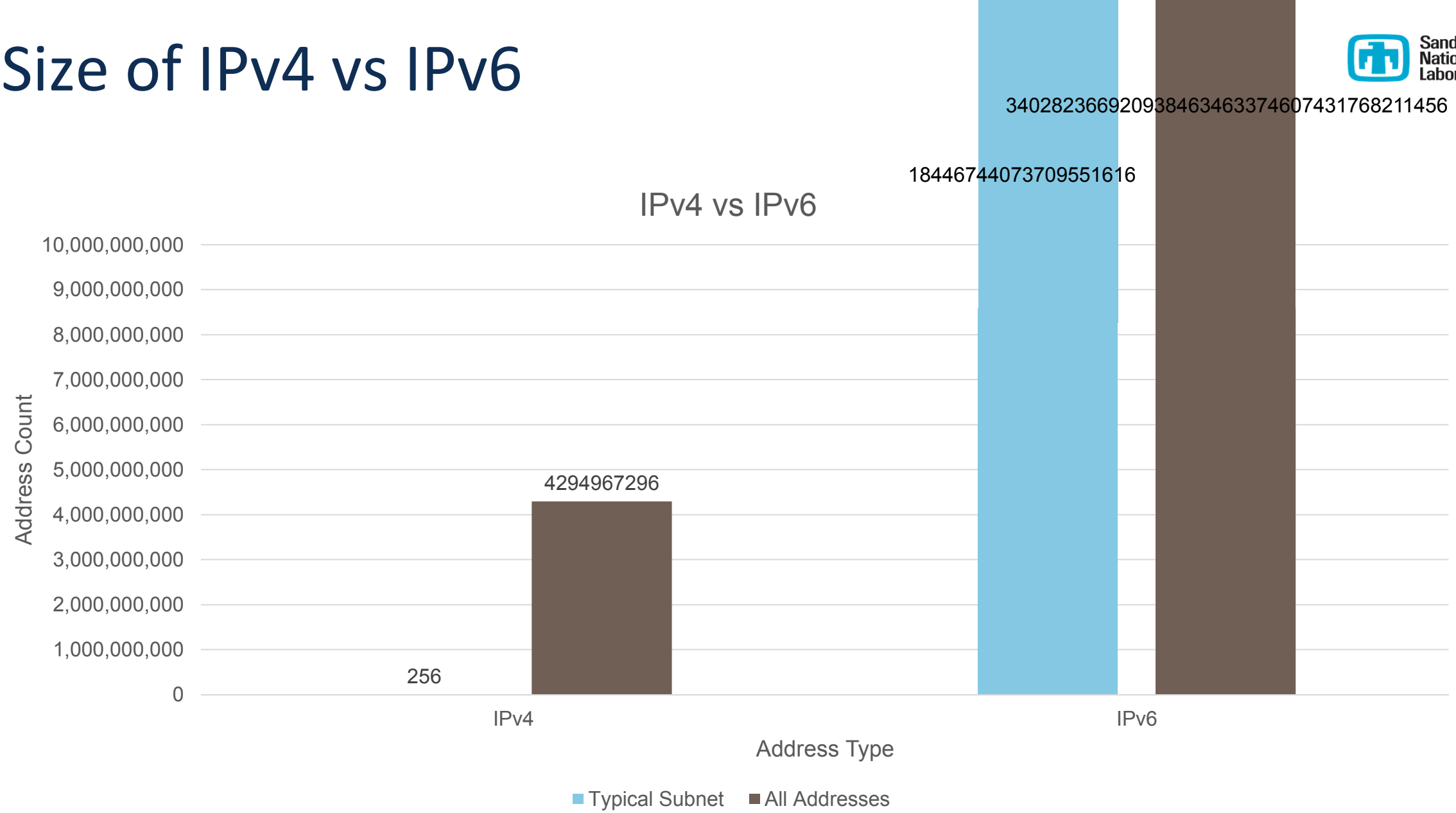
By Andy0101 (talk) - Own work, CC BY 3.0, https://commons.wikimedia.org/w/index.php?curid=10700518

IP Addresses are just numbers

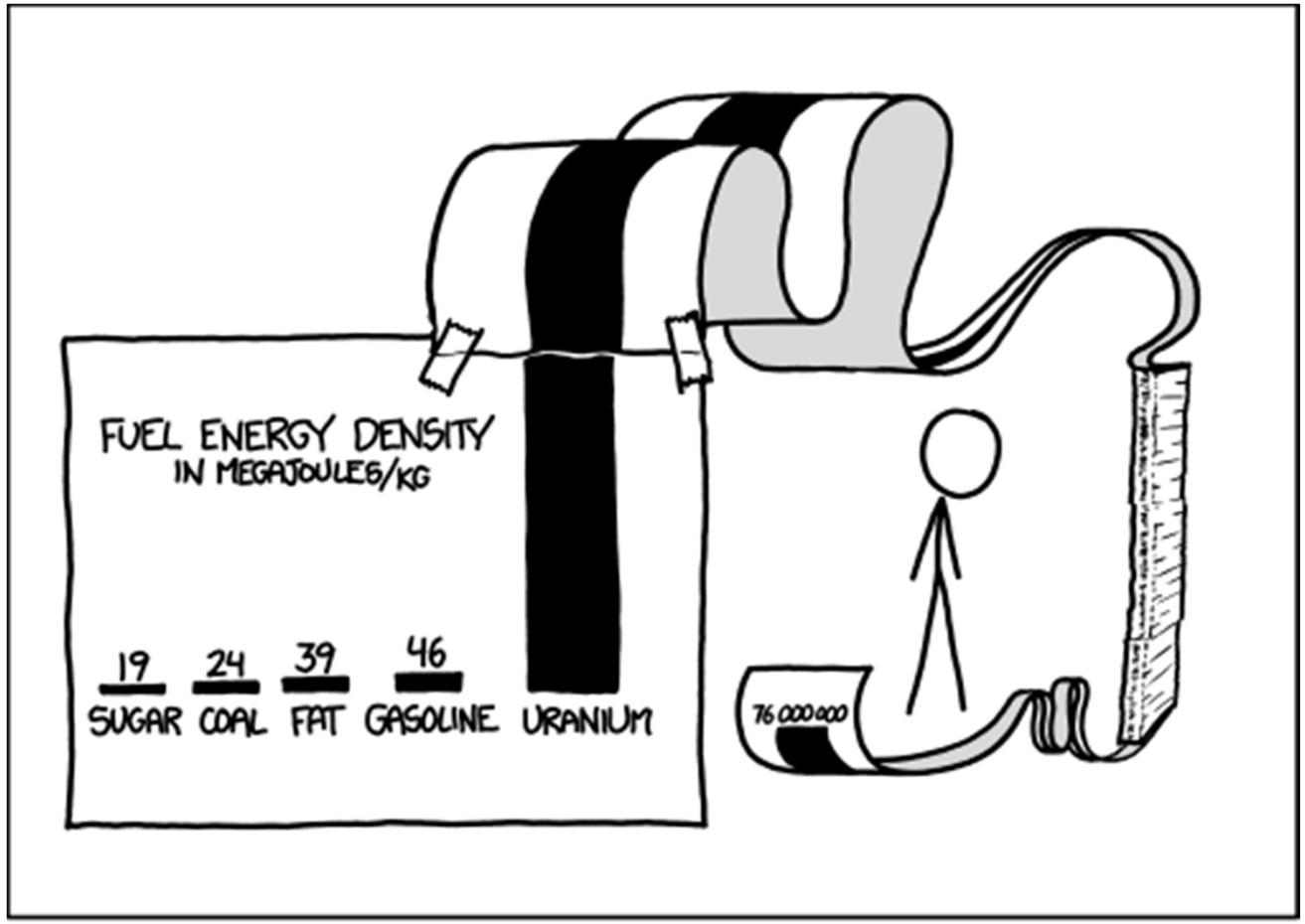
Sandia National Laboratories

- IPv4
  - 2^{32}
  - ~2G (Giga)
  - 4,294,967,296 possibilities
  - Ranges from 0.0.0.0 to 255.255.255.255
- IPv6
  - 2^{128}
  - ~$3x10^{14}$Y (Yotta).. way bigger than 9Exa
  - 340,282,366,920,938,463,463,374,607,431,768,211,456 possibilities
  - Ranges from :: to ffff:ffff:ffff:ffff:ffff:ffff:ffff:ffff

Size of IPv4 vs IPv6



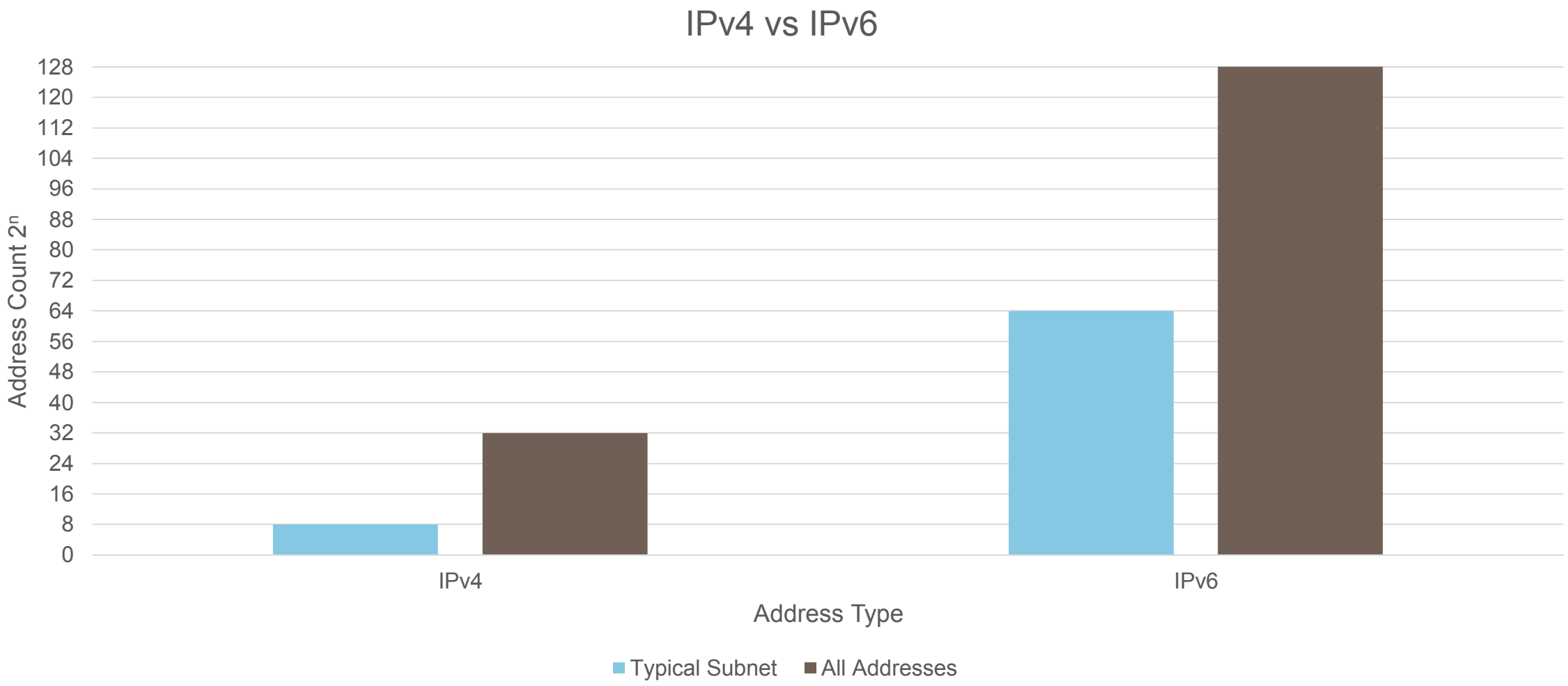
Log Scales are for Quitters



SCIENCE TIP: LOG SCALES ARE FOR QUITTERS WHO CAN'T FIND ENOUGH PAPER TO MAKE THEIR POINT *PROPERLY.*

Log Scale https://xkcd.com/1162/

Fine, have your Log_2 scales

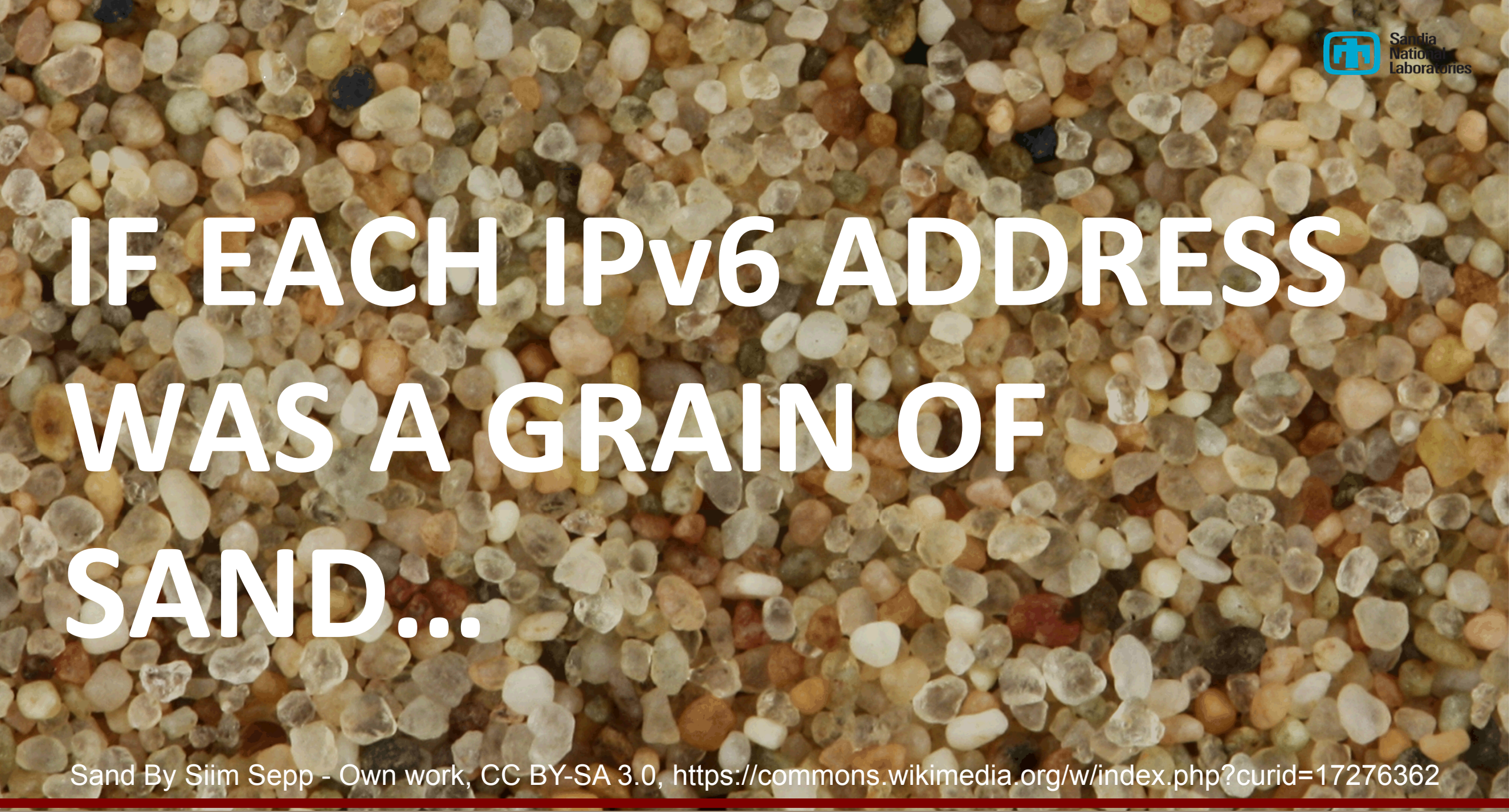


Sandia National Laboratories

BUT HOW BIG IS THAT REALLY?

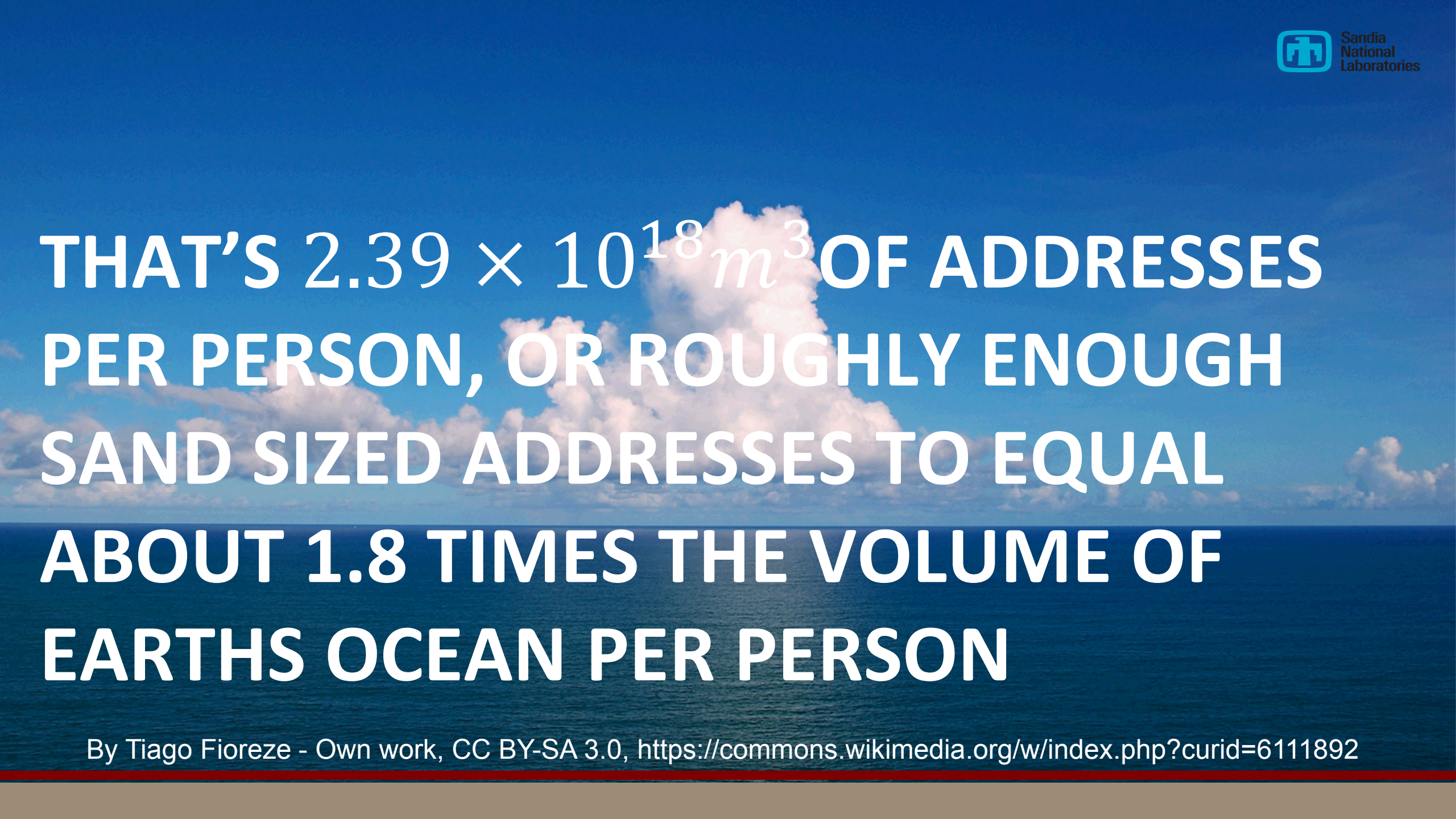
Let’s divvy up those addresses

Crowd By James Cridland - Own work, CC BY-SA 2.0, https://www.flickr.com/photos/jamescridland/6134456810



Sand By Siim Sepp - Own work, CC BY-SA 3.0, https://commons.wikimedia.org/w/index.php?curid=17276362

Sandia National Laboratories


Sandia National Laboratories
THAT'S $2.39 \times 10^{18} m^3$ OF ADDRESSES PER PERSON, OR ROUGHLY ENOUGH SAND SIZED ADDRESSES TO EQUAL ABOUT 1.8 TIMES THE VOLUME OF EARTHS OCEAN PER PERSON
By Tiago Fioreze - Own work, CC BY-SA 3.0, https://commons.wikimedia.org/w/index.php?curid=6111892

Sandia National Laboratories

AT SAND SCALE ALL IPv6 ADDRESSES WOULD TAKE THE VOLUME OF ABOUT 12 SOLS

By NASA/SDO/AIA/Goddard Space Flight Center - http://www.nasa.gov/mission_pages/rbsp/news/third-belt.html, Public Domain, https://commons.wikimedia.org/w/index.php?curid=24922453

Sandia National Laboratories

CONVERSELY, ALL THE IPv4 ADDRESSES IN THIS SAND SCALE WOULD SLIGHTLY OVER FILL AN OIL DRUM

Sandia National Laboratories

COMPOSITION

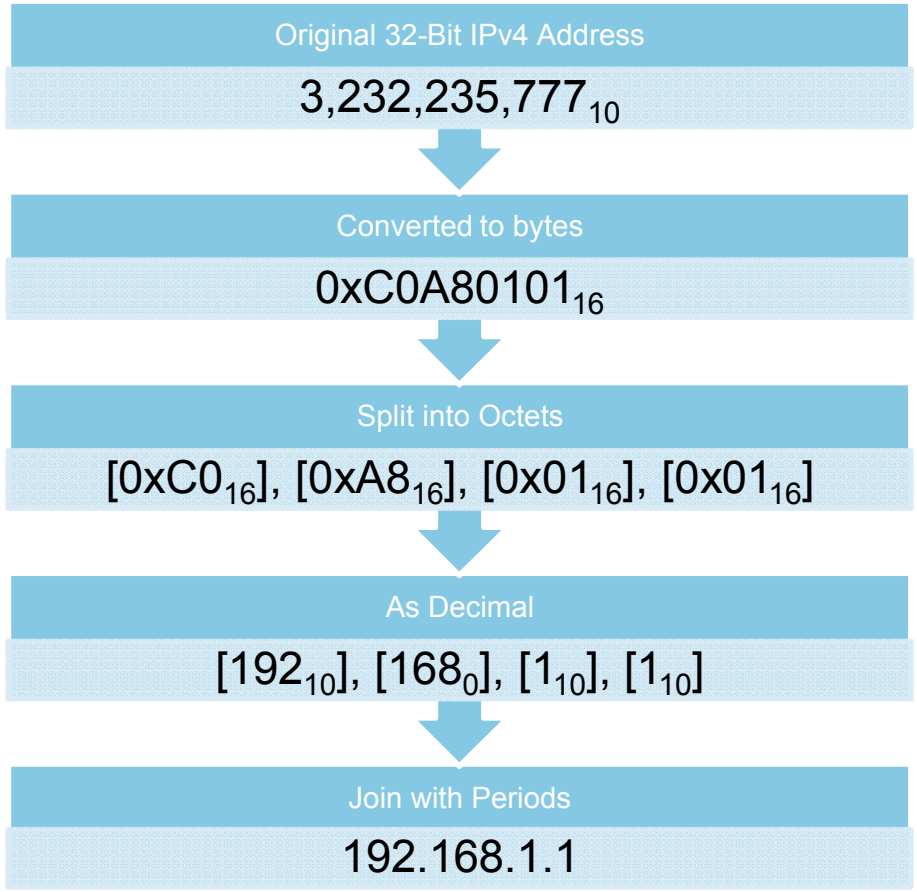
How is IP Address formed

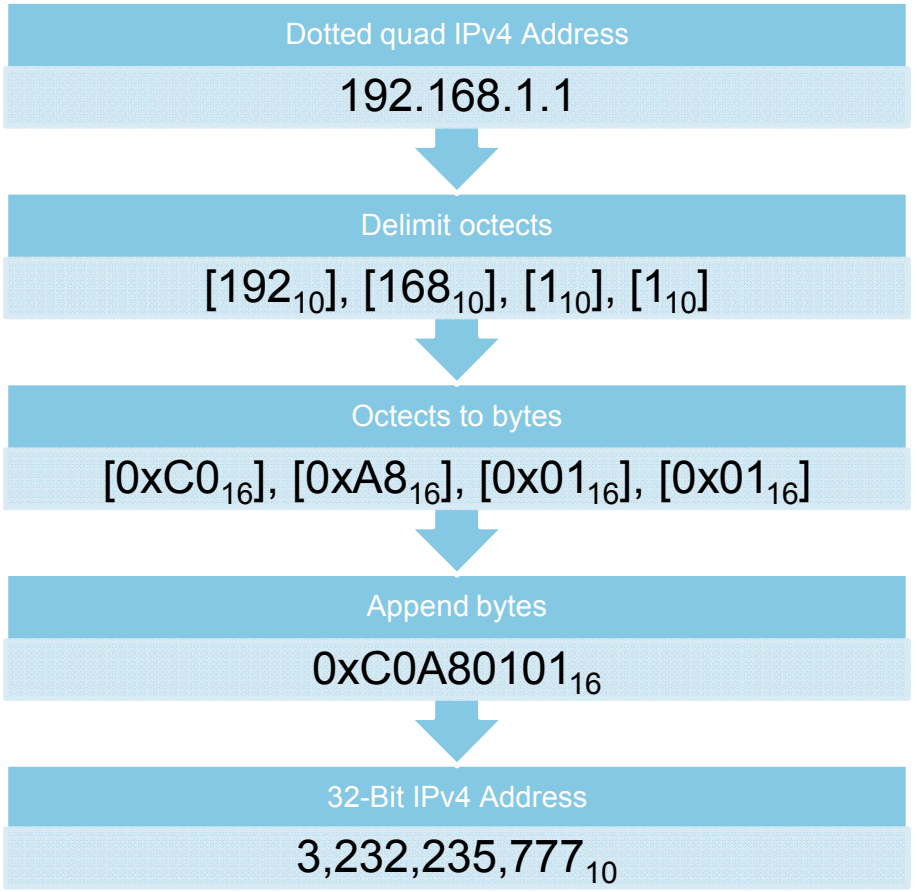
Format of IPv4

Sandia National Laboratories

IPv4 Address Equality		
$3{,}232{,}235{,}777_{10}$ Unsigned Decimal	$0xC0A80101_{16}$ Hexadecimal big-endian	192.168.1.1 Dotted Quad

Converting IPv4





Format of IPv6

Sandia National Laboratories

- IPv6 Hexadecimal Notation
  - Hiding zeros, sometimes, to make 128-bit numbers more human readable, sometimes

IPv6 Address Equality			
$334,965,470,149,915,403,112,039,664,316,435,185,886_{10}$ Unsigned Decimal	0xFC0000C000FFEE15600D00000000C0DE16 $\text{Hexadecimal}_{\text{big-endian}}$	fc00:00c0:00ff:ee15:600d:0000:0000:c0de Expanded From	fc00:c0:ff:ee15:600d::c0de Hexadecimal notation

Take a good look at IPv6

Sandia National Laboratories

Un-collapsed

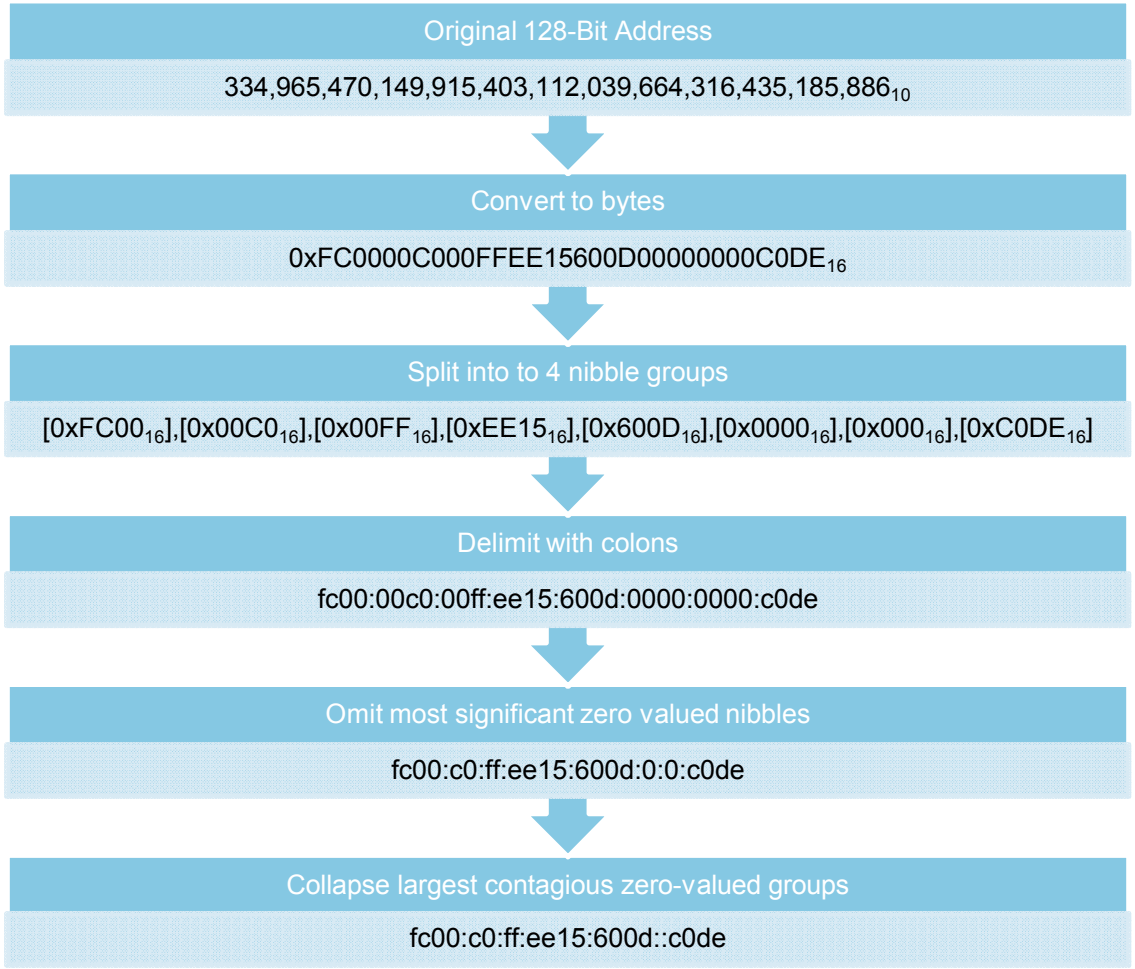
fc00:00c0:00ff:ee15:600d:0000:0000:c0de

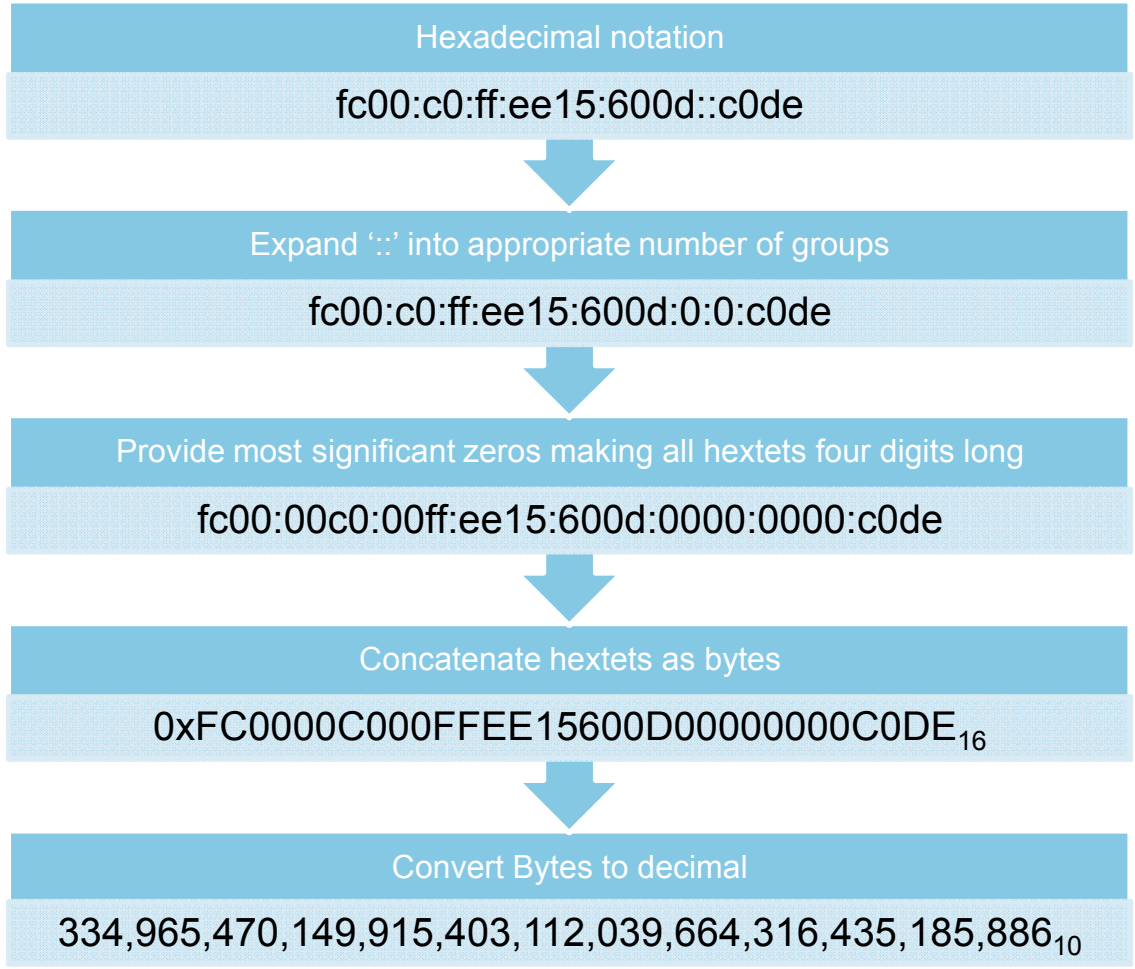
Collapsed

fc00:c0:ff:ee15:600d::c0de

Converting IPv6

Sandia National Laboratories





Sub Networks a.k.a. Subnetting

Sandia National Laboratories

- Components
  - Network Bits and Host Bits
  - Boundaries
- Formats
  - Netmasking
  - ~~Classful Networks~~
  - CIDR

Network Bits and Host Bits

- Network Bits – Specifies the sub network
- Host Bits – Specifies the range of allowed hosts on a network

Host Bits & Network Bits	
host bits = bit depth - network bits	network bits = bit depth - host bits

Network Bits and Host Bits

Sandia National Laboratories

Host bits (n)	IPv4 Network Bits ($32-n$)	IPv6 Network Bits ($128-n$)	Addresses in Subnet as 2^n	Addresses in Subnet
0	32	128	2^0	1
1	31	127	2^1	2
2	30	126	2^2	4
3	29	125	2^3	8
4	29	124	2^4	16
5	27	123	2^5	32
6	26	122	2^6	64
7	25	121	2^7	128
8	24	120	2^8	256
9	23	119	2^9	512
⋮				
32 (IPv4 Maximum)	0	96	2^{32}	4,294,967,296
⋮				
128 (IPv6 Maximum)	not applicable	0	2^{128}	$\sim 3.402 \times 10^{38}$

Bitmasking (Netmasking, well sorta)

Sandia National Laboratories

- A Netmask is just a IPv4 encoded Bitmask
- Operations
  - AND an address with a Network Bitmask to get Network Bits
  - XOR an address with a Network Bitmask to get The Last Address

Bitmasking

Sandia National Laboratories

Network bits	Host bits	Netmask	Mask as integer	Bitmask	Address Count
0	32	0.0.0.0	0	0000 0000 0000 0000 0000 0000 0000 0000	4294967296
1	31	128.0.0.0	2147483648	1000 0000 0000 0000 0000 0000 0000 0000	2147483648
2	30	192.0.0.0	3221225472	1100 0000 0000 0000 0000 0000 0000 0000	1073741824
3	29	224.0.0.0	3758096384	1110 0000 0000 0000 0000 0000 0000 0000	536870912
4	28	240.0.0.0	4026531840	1111 0000 0000 0000 0000 0000 0000 0000	268435456
5	27	248.0.0.0	4160749568	1111 1000 0000 0000 0000 0000 0000 0000	134217728
6	26	252.0.0.0	4227858432	1111 1100 0000 0000 0000 0000 0000 0000	67108864
7	25	254.0.0.0	4261412864	1111 1110 0000 0000 0000 0000 0000 0000	33554432
8	24	255.0.0.0	4278190080	1111 1111 0000 0000 0000 0000 0000 0000	16777216
9	23	255.128.0.0	4286578688	1111 1111 1000 0000 0000 0000 0000 0000	8388608
10	22	255.192.0.0	4290772992	1111 1111 1100 0000 0000 0000 0000 0000	4194304
11	21	255.224.0.0	4292870144	1111 1111 1110 0000 0000 0000 0000 0000	2097152
12	20	255.240.0.0	4293918720	1111 1111 1111 0000 0000 0000 0000 0000	1048576
13	19	255.248.0.0	4294443008	1111 1111 1111 1000 0000 0000 0000 0000	524288
14	18	255.252.0.0	4294705152	1111 1111 1111 1100 0000 0000 0000 0000	262144
15	17	255.254.0.0	4294836224	1111 1111 1111 1110 0000 0000 0000 0000	131072
16	16	255.255.0.0	4294901760	1111 1111 1111 1111 0000 0000 0000 0000	65536
17	15	255.255.128.0	4294934528	1111 1111 1111 1111 1000 0000 0000 0000	32768
18	14	255.255.192.0	4294950912	1111 1111 1111 1111 1100 0000 0000 0000	16384
19	13	255.255.224.0	4294959104	1111 1111 1111 1111 1110 0000 0000 0000	8192
20	12	255.255.240.0	4294963200	1111 1111 1111 1111 1111 0000 0000 0000	4096
21	11	255.255.248.0	4294965248	1111 1111 1111 1111 1111 1000 0000 0000	2048
22	10	255.255.252.0	4294966272	1111 1111 1111 1111 1111 1100 0000 0000	1024
23	9	255.255.254.0	4294966784	1111 1111 1111 1111 1111 1110 0000 0000	512
24	8	255.255.255.0	4294967040	1111 1111 1111 1111 1111 1111 0000 0000	256
25	7	255.255.255.128	4294967168	1111 1111 1111 1111 1111 1111 1000 0000	128
26	6	255.255.255.192	4294967232	1111 1111 1111 1111 1111 1111 1100 0000	64
27	5	255.255.255.224	4294967264	1111 1111 1111 1111 1111 1111 1110 0000	32
28	4	255.255.255.240	4294967280	1111 1111 1111 1111 1111 1111 1111 0000	16
29	3	255.255.255.248	4294967288	1111 1111 1111 1111 1111 1111 1111 1000	8
30	2	255.255.255.252	4294967292	1111 1111 1111 1111 1111 1111 1111 1100	4
31	1	255.255.255.254	4294967294	1111 1111 1111 1111 1111 1111 1111 1110	2
32	0	255.255.255.255	4294967295	1111 1111 1111 1111 1111 1111 1111 1111	1

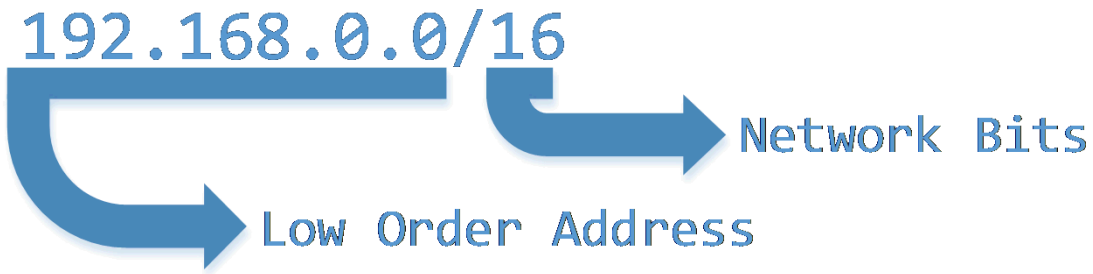
~~Classful Networks~~

Sandia National Laboratories

- No, you're wrong
- Stop
- Don't do this
- Go away
- This is not the slide deck you are looking for

Classless Inter-Domain Routing (CIDR)

- Fixes issues with Classful Networks
- Valid for IPv4 and IPv6
- Specifies start address and number of bits must be static on the left



Sandia National Laboratories

DISCUSSING IMPLEMENTATION

What to keep in mind when losing your mind

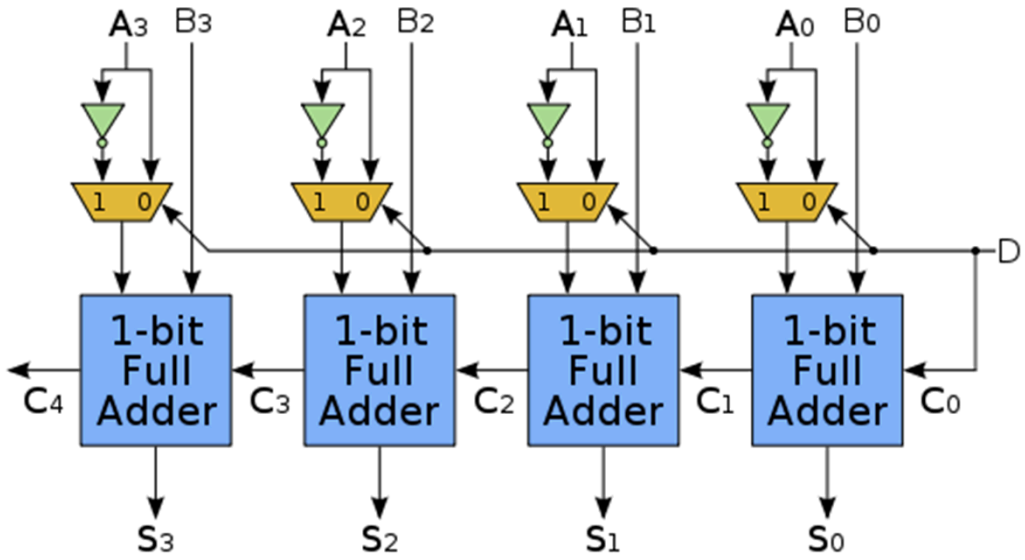
Implementation Details

Sandia National Laboratories

- Sign parity
- Endianness
- Programmatic Mathematics
  - 32-bit Integers to IPv4
  - 128-bit Integers to IPv6

Sign Parity

- Not all bits are the same
- IP Addresses are unsigned by nature



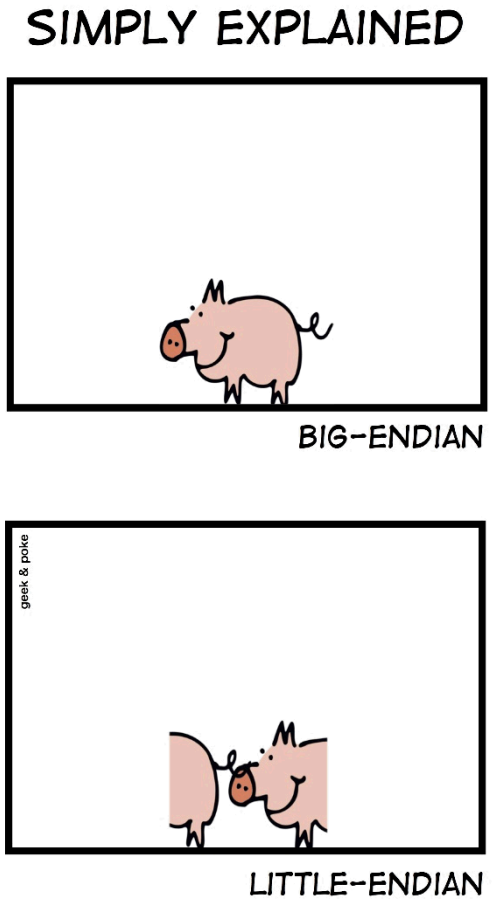
Endianness - it's all about word order

Sandia National Laboratories

- Big Endian
  - Network order
- Little Endian
  - System Bus Order (typically)

Endian Equality		
$1{,}123{,}029{,}93210_{10}$ Decimal	$[0x42_{16}],[0xF0_{16}],[0x13_{16}],[0xAC_{16}]$ Big-endian	$[0xAC_{16}],[0x13_{16}],[0xF0_{16}],[0x42_{16}]$ Little-endian

Endianness



http://geekandpoke.typepad.com/geekandpoke/2011/09/simply-explained-1.html

Sandia National Laboratories

PROGRAMMATIC MATHEMATICS

In a pseudo C# like environment

Sandia National Laboratories

THIS SHOULD BE EASY, THIS PROBLEM SEEMS WELL UNDERSTOOD…

Sandia National Laboratories

…YOUR COMPUTER DOESN'T DO 128-BIT MATH

Even if it does, chances are your programming language doesn't

What needs to be done – IP Addresses

Sandia National Laboratories

- Treat as a “First-class citizen”
- Math - Increment / Decrement / Mathematically Compare
- Know Address Family (IPv4 vs IPv6)
- Put data to rest / read data out
- User/Developer friendly interaction
  - Parse
  - Output
- Logical Composition

What needs to be done – Subnets

Sandia National Laboratories

- Treat as “First-class citizen”
- Basic Math – Mathematically Comparison
- Know Address Family (IPv4 vs IPv6)
- Put data to rest / read data in
- User/Developer friendly interaction
  - Parse
  - Output
- Set Mathematics
  - Expand
  - Contract
  - Divide
  - Combine
  - Membership Introspection
- Logical Composition
  - Arbitrary Range Best fit
  - Consecutive fitting of Arbitrary Range

What we need

Sandia National Laboratories

We need a data type that is:

- Big-endian
- Capable of bitwise arithmetic
- Integer Based
- Unsigned
- Variable bit depth up to 128-Bit
- ~~Willing to make me coffee in the morning~~

What do we have

Sandia National Laboratories

- Common data types
  - Integers
  - Bytes
- C# data types
  - IPAddress - mostly just some special logic around bytes

Integers (BigInteger)

Sandia National Laboratories

Integers CAN

- Basic Arithmetic for large numbers
- Integer Based
- Variable bit depth
- Be treated as numbers

Integers CAN'T

- Big-endian
- Boolean Arithmetic
- Be unsigned

Byte array

Sandia National Laboratories

Byte Arrays CAN

- Boolean Arithmetic
- Convert between endianness
- Variable bit depth

Byte Arrays CAN'T

- Basic Arithmetic for large numbers
- Integer Based
- Be treated as numbers

IP Address Type

Sandia National Laboratories

IP Addresses CAN

- Big-endian byte order
- Look pretty
- Parse from IP Address style strings
- Variable bit depth (32-bit and 128-bit)

IP Addresses CAN'T

- Basic Arithmetic
- Boolean Arithmetic
- Be treated as numbers

LETS MASH SOME THINGS TOGETHER

By our powers combined

byte[] Operations– Basic Building Blocks

Sandia National Laboratories

- Bitwise operations
  - NOT, AND, OR, XOR
  - Bit-shifting
  - Bit Padding (to a lesser degree)
- Convert back and forth between
  - IP Address
  - BigInteger

BigInteger Operations– Basic Building Blocks

Sandia National Laboratories

- Mathematical
  - Treat as unsigned
  - Byte Padding
- Convert back and forth between
  - IP Address
  - Byte[]of appropriate endianness
  - Strings of appropriate base

IPAddress Operations– Basic Building Blocks

Sandia National Laboratories

- Mathematical
  - Increment / Decrement
  - Mathematic equality and comparison
- Convert back and forth between
  - Netmasks
  - Special Case Parsing
  - Suitable Integers
  - Byte[] of appropriate endianness
  - Strings of appropriate base and Formats

Sandia National Laboratories

WHAT ABOUT SUBNETS?

Then a miracle occurs…

Sandia National Laboratories

AT THIS POINT A SUBNET IS JUST TWO NUMBERS

The first address, and the last address

Subnet – Basic Math

Sandia National Laboratories

- Comparison
  - Size
    - Last address minus First address
  - Ordering
    - Value of $Address_1$ against Value of $Address_2$

Subnet – Set Math

Sandia National Laboratories

- Expand
  - High address is increased 2^n orders of magnitude
- Contract
  - High address is decreased 2^n orders of magnitude
- Divide
  - New subnet begins at half way point of old subnet, old subnet ends at halfway point
- Membership Introspection
  - Address membership
    - Compare address to high and low address
  - Subnet membership
    - Subnet high and low between high and low

Subnet – Logical Composition

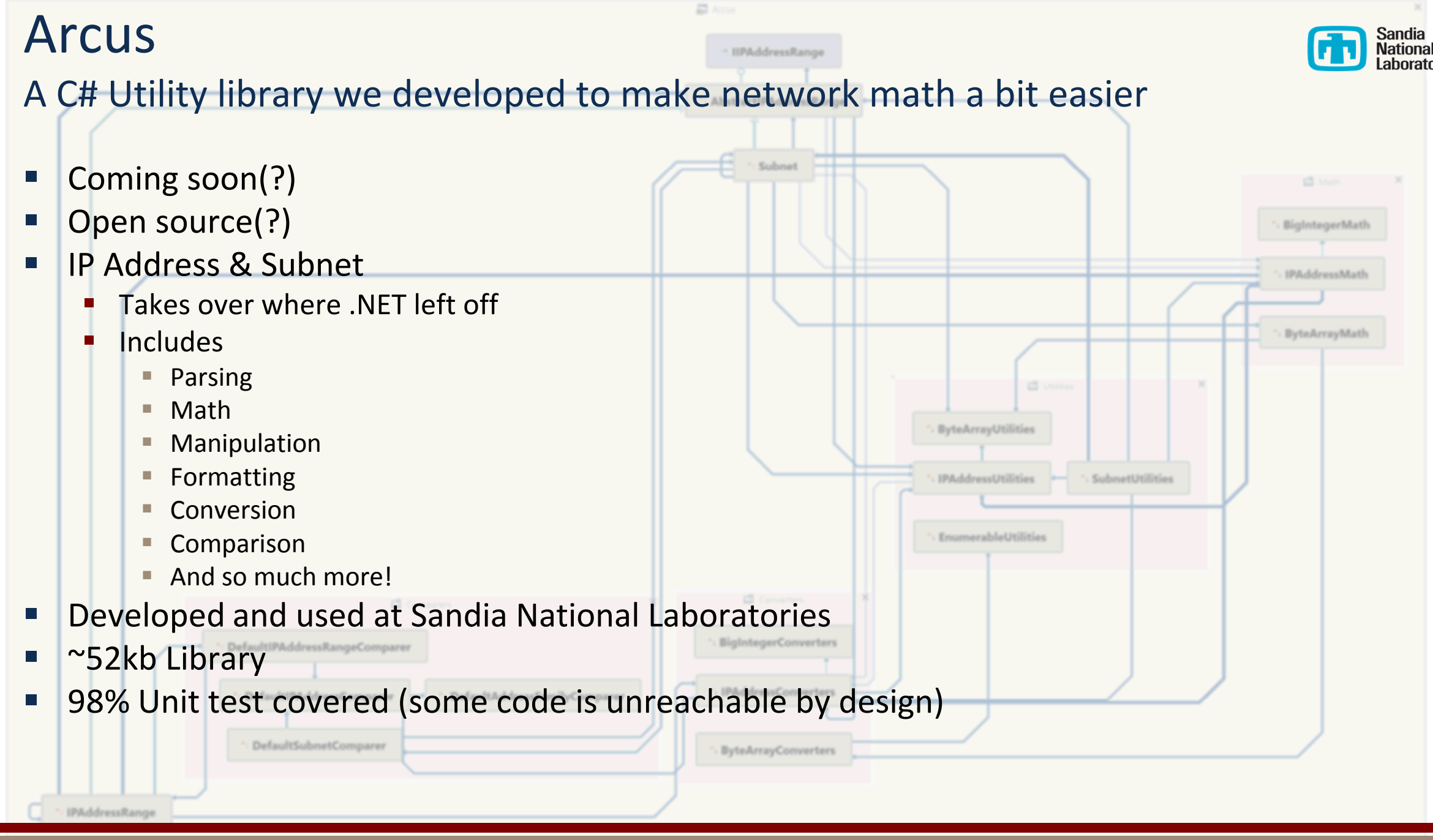
Sandia National Laboratories

- Arbitrary Range Best fit
  - AND the low and high address to find where the subnet starts
  - XNOR the low and the high and count (hamming) the consecutive set bits for network bits
    - Bitmask is 2^{32}-$2^{32\text{-network bits}}$ for IPv4
    - Bitmask is 2^{128}-$2^{128\text{-network bits}}$ for IPv6
  - XNOR the low address by the bitmask for the high address
- Consecutive fitting of Arbitrary Range
  - Binary Search: Recursively divide range by two until each sub range is filled

Arcus

A C# Utility library we developed to make network math a bit easier

- Coming soon(?)
- Open source(?)
- IP Address & Subnet
  - Takes over where .NET left off
  - Includes
    - Parsing
    - Math
    - Manipulation
    - Formatting
    - Conversion
    - Comparison
    - And so much more!
- Developed and used at Sandia National Laboratories
- ~52kb Library
- 98% Unit test covered (some code is unreachable by design)



Sandia National Laboratories

~~YOUR COMPUTER DOESN'T DO 128-BIT MATH~~

Now it does

Sandia National Laboratories

PARTIAL RELATIONAL DATABASE IMPLEMENTATION

SQL ˈsiːkwəl/; (n): A four letter word so repugnant that it only requires three.

Options for data at rest

Sandia National Laboratories

The Breaking

- Integer (int and bigint)
  - Sign issues
  - Too small
- Split data up
  - 8 columns for int
  - 4 columns for bigint
- Varchar
  - No math
  - Cannot order well

The Bad?

- Binary
  - Overcomes issues with other data types
  - Hard to read for IPv4
    - Until you get used to it

IP address table

Sandia National Laboratories

address : varbinary(16)	address_family : varchar(16)
0xFC0000C000FFEE15600D00000000C0DE	InterNetworkV6
0xC0A80101	InterNetwork

Subnet table

Sandia National Laboratories

begin_address : varbinary(16)	end_address : varbinary(16)	address_family : varchar(16)
0xFC000000000000000000000000000000	0xFC0000FFFFFFFFFFFFFFFFFFFFFFFF	InterNetworkV6
0xC0A80101	0xC0A801FF	InterNetwork

Address bytes to string

Sandia National Laboratories

- No one wants to look at `0xC0A80101`
  - So we pull it apart
    - `substring(0xC0A80101, 1, 1) = C0`
    - `CAST(substring(0xC0A80101, 1, 1) AS int) = 192`
    - `CAST(CAST(substring(0xC0A80101, 1, 1) AS int) AS nvarchar(3)) + '.' = 192.`
- Lather rinse and repeat until we have `192.168.1.1`
- Nor does anyone want to look at `0xFC0000C000FFEE15600D00000000C0DE`
  - Thankfully you can fizz buzz your way to `FC00:00C0:00FF:EE15:600D:0000:0000:C0DE`

Address string to bytes

Sandia National Laboratories

- Splitting up 192.168.1.1 into octets
  - Bit shift each octet by an octet
    - `192 * 256 * 256 * 256 = 0xC00000`
    - `168 * 256 * 256 = 0x00A80000`
    - `1 * 256 = 0x00000100`
    - `1 = 0x00000001`
  - Add bit shifted octets together to get `0xC0A80101`
- Remove colons and cast `FC00:00C0:00FF:EE15:600D:0000:0000:C0DE`

Bend SQL to your will

Sandia National Laboratories

- Find next available integer within range that is not present in existing integer subset
  - given a range x -> y of unsigned integers
    - where x and y are both in the range 0 -> 2^n
    - and n is 0 -> 32 (or 64 in IPv6 case)
  - find the minimum available value
    - not equal to x or y
      - that is not in an existing set
      - where existing sets are arbitrary subsets of x -> y

What We've Learned

Sandia National Laboratories

- The scale of IPv6 is really, really big
  - To a somewhat silly degree
- IP Addresses are just numbers
  - We just need to know how to treat them right
- Subnets are just number ranges
  - There is a right way, and a wrong way to define these ranges
- Computers and languages don't want to do IPv6 mathematics
  - You can make it work if we divide and conquer
  - Or we may have a solution for you (Arcus)
- SQL wasn't really meant to deal with big numbers
  - We made it do it anyhow

Sandia National Laboratories

QUESTIONS?

If not, we will be leading the session in a lively rendition of the popular networking song "2^{128} IPv6 Addresses on the wall"

Andrew Steele
Andrew.Steele@sandia.gov
@ahsteele

Robert H. Engelhardt
Robert.Engelhardt@sanida.gov
@rheone

Sandia National Laboratories

QUESTIONS?

In fairness, we warned you

Andrew Steele
Andrew.Steele@sandia.gov
@ahsteele

Robert H. Engelhardt
Robert.Engelhardt@sanida.gov
@rheone