Networking Basics: IP, DNS, URL, MIME

Lecture 11

Internet Protocol (IP) Addresses

- A unique 32-bit number
  - Assigned to device connected to internet
  - An address for delivery of packets
- Written in *dotted-decimal* notation
  - Divided into 4 fields separated by "."
  - Each field is 8 bits, ie 0-255 decimal

    10100100011010110111101100000110

    10100100.01101011.01111011.00000110

    164.107.123.6
- Some are reserved: eg, 127.0.0.1

Abstract Value vs Encoding

- Abstraction: 32-bit integer value
- Encodings
  - Dotted decimal
  - Dotted hex
  - Dotted octal
  - Hexadecimal
  - Decimal
  - Binary
  - Etc…
- Recall: abstraction, representation, *correspondence relation*

Address Space

- Organizations are allocated blocks of contiguous address to use
- 32 bits means 4 billion addresses
  - Population of the earth: 7 billion
  - Not enough addresses to go around!
- The end is predictable
  - Techniques like NAT developed to help
- In fact, the end has come!
  - Feb 2011: Last block was allocated

IPv6

- 128 bits
  - ~10^{40} addresses; we're good for a while
  - A growing fraction of IP traffic
    GoogleIPv6 statistics
- Recommended format (canonical):
  - Divide into 8 fields separated by ":"
  - Each field is 4 hex digits (0-FFFF), ie 16 bits
  - Omit *leading* 0's in a field
  - If there are *consecutive* fields with value 0, compress them as "::"
  - Compress *at most one* such set of 0's
    - Otherwise encoding could be ambiguous
    - Compress the longest sequence

Canonical Format: Uniqueness

2001:0db8:0000:0000:0000:ff00:0042:8329

2001:0db8:0000:0000:0000:ff00:0042:8329

2001:db8:0:0:0:ff00:42:8329

2001:db8:0:0:0:ff00:42:8329

2001:db8::ff00:42:8329

Domain Names

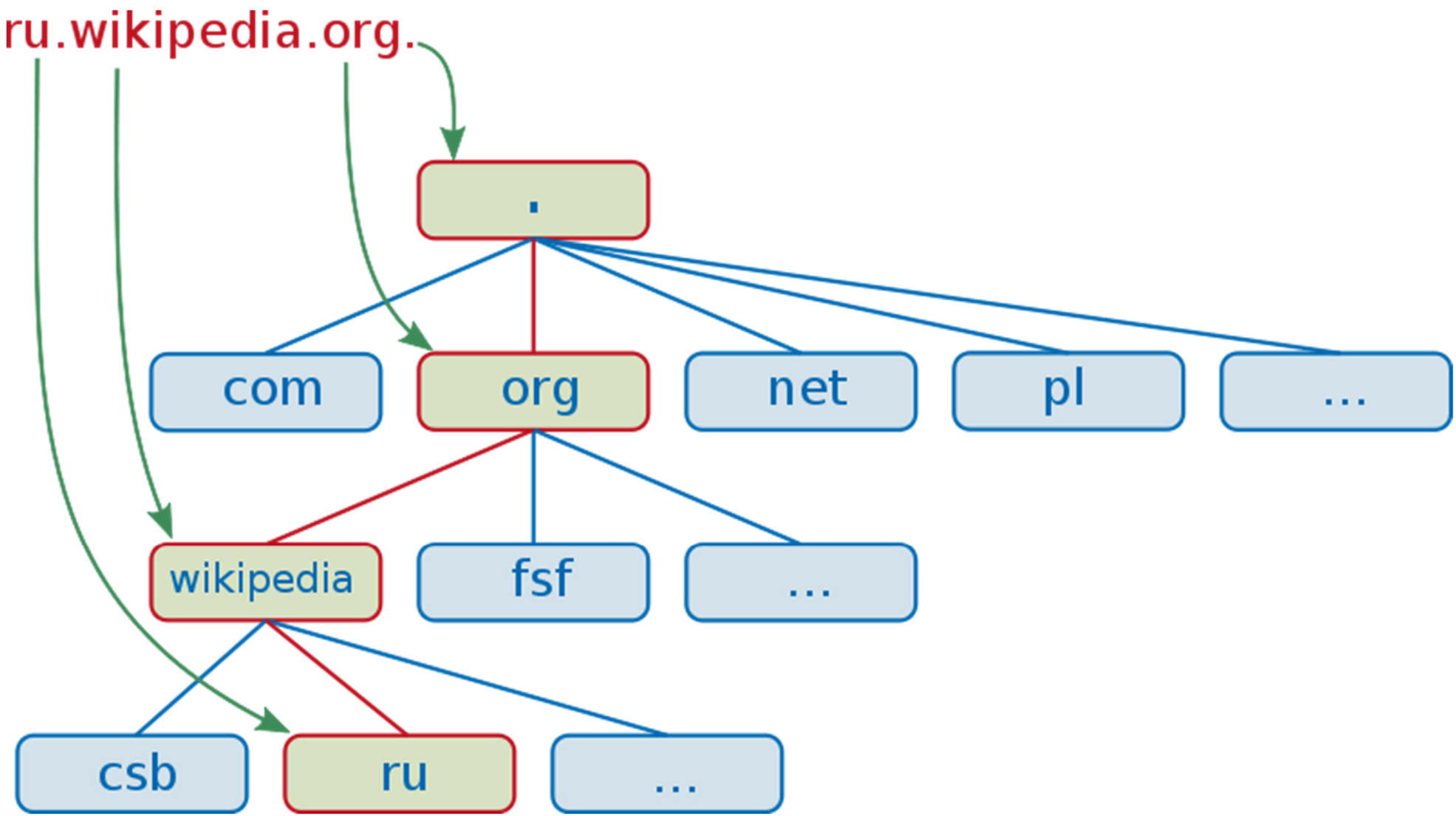
- String corresponds to an IP address
  - `web.cse.ohio-state.edu` is easier than `164.107.123.6`
- Case insensitive: Lower-case standard
- A partial map (almost)
  - DNS maps lower-case strings → IP addresses
  - Multiple strings can map to same address!
  - Some strings map to multiple addresses (unusual)!

Domain Name Hierarchy

- Separated by .'s
  - Don't confuse with dotted decimal!
- Right-to-left hierarchy
  - Top-level domain is right-most field
    - **edu**, **com**, **net**, **gov**, countries (**ca**, **it**, …)
  - Second-level domain to its left
  - Then third, fourth, etc, no limit

    www.sos.state.oh.us
- Hostname + Domain Name = Fully Qualified Domain Name (FQDN)

  stdlinux.cse.ohio-state.edu


ru.wikipedia.org.
.
com
org
net
pl
...
wikipedia
fsf
...
csb
ru
...

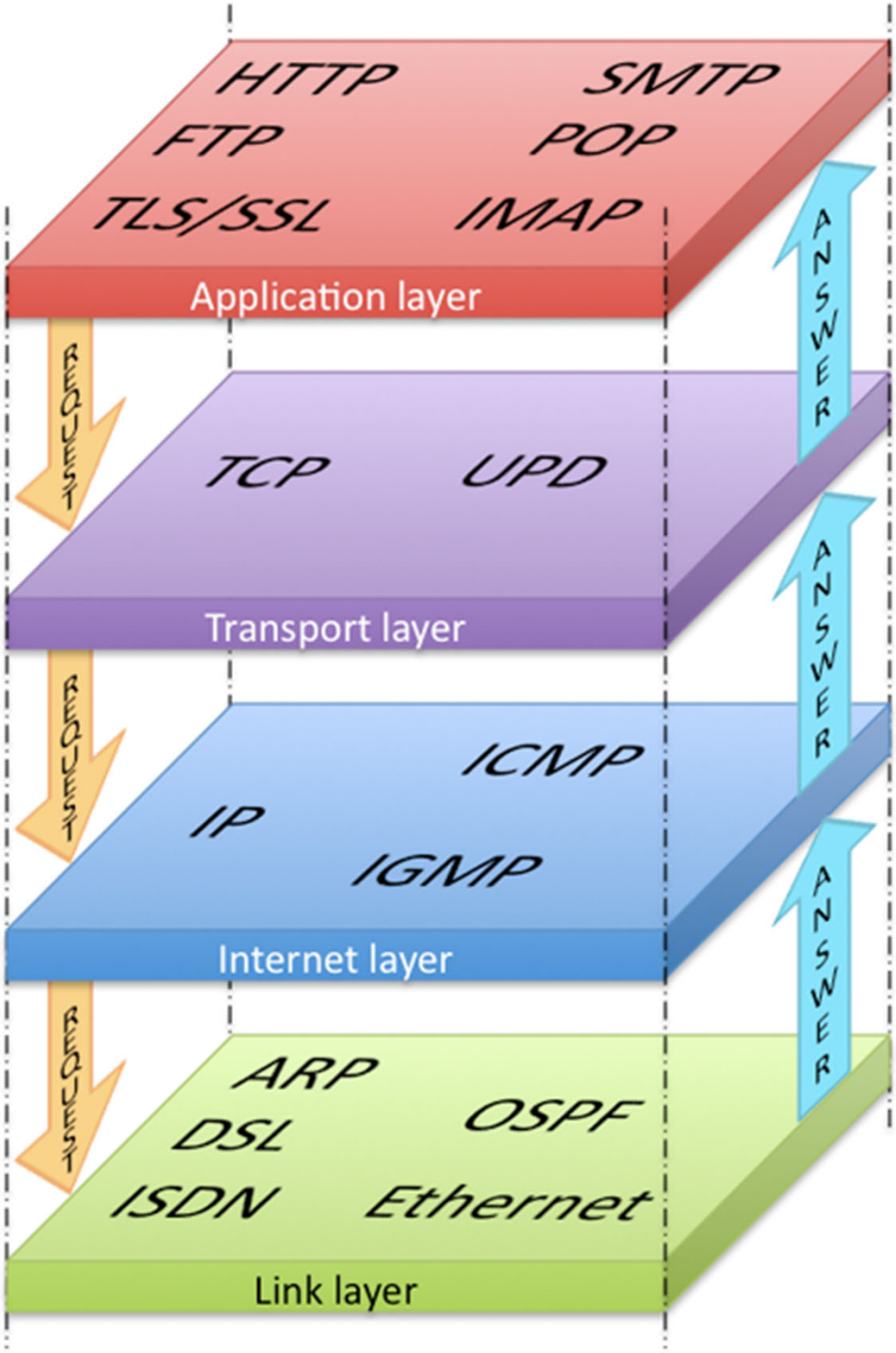
Name Servers

- Act as a phonebook for lookup
- Client view:
  - Given a FQDN, return IP address
  - Partial map: FQDNs → IP addresses
  - See host, whois
- Implementation view:
  - Hierarchical by domain
  - Local caching for recently retrieved items
- Command line tools

```
$ host web.cse.ohio-state.edu
# web.cse.ohio-state.edu has address
      164.107.129.176
$ whois osu.com
```

Protocols

- Systematic ordering of messages
  - Phone rings
  - Callee answers by saying “Hello”
  - Caller answers by saying “Hello”
- Different protocols use different messages, different sequencing, etc
  - In Italy, callee answers by saying “*Pronto*”


HTTP
SMTP
FTP
POP
TLS/SSL
IMAP
Application layer
REQUEST
ANSWER
TCP
UPD
Transport layer
REQUEST
ANSWER
ICMP
IP
IGMP
Internet layer
REQUEST
ANSWER
ARP
OSPF
DSL
ISDN
Ethernet
Link layer

Network Layering: Abstraction

- One protocol is built on top of another
  - Application level: FTP, HTTP, SSH, SMTP, TELNET
  - Transport: TPC, UDP
  - Internet: IP
- Each protocol assumes certain behavior from layer below
  - IP routes packets to destination (unreliable)
  - TCP creates a reliable, in-order channel
  - HTTP delivers web pages

Network Ports

- A single host has many ports
- Application-level protocols have default port
  - ftp -> 20
  - http -> 80
  - imap ->143
  - ssh -> 22
  - smtp -> 25
  - telnet-> 23
- A “web server” is just a program, running, waiting, listening for a call (on port 80)
  - See telnet

URL

- Uniform Resource Locator

  scheme://FQDN:port/path?query#fragment

- Schemes include http, ftp, mailto, file…
  - Case insensitive, but prefer lower case
- Port is optional (each scheme has default)
  - 80 for http
- Variety of formats, depending on scheme

  http://www.osu.edu/news/index.php

  ftp://doe@ftp.cse.ohio-state.edu

  mailto://brutus.1@osu.edu

- FQDN is case insensitive, prefer lower case

Document Root

- Web server configured to serve documents from a location in file system
  - "document root": `/class/3901`
  - File: `/class/3901/labs/lab2.html`
  - URL: `http://www.cse.osu.edu/labs/lab2.html`
- Slashes in path should be for server's OS (but forward slashes are common)
- Virtual servers: multiple doc roots
- Proxy servers: remote doc roots

Encoding (and Decoding)

- A single value can be viewed at two levels, eg:
  - **HELLO**
  - .... . .-.. .-.. ---
- Different uses: reading vs transmission
- Different alphabets (letters vs dot-dash) and/or requirements
  - Eg. Message has only upper case letters
- Encoding/decoding is the translation between these levels
  - c.f. encrypting/decrypting
- Abstract value vs concrete representation
  - Correspondence maps between the two

Example: URL Encoding

- Invariant on abstract value (constraint)
  - Reserved metacharacters (`;`, `:`, `&`, `#`, `@`…)
- Invariant on encoding (convention)
  - Small set of valid characters, others (eg space, `~`, newline…) are not allowed
- So some characters in abstract value are encoded as `%hh` (ASCII code in hex)
  - `%3B` for `;`, `%40` for `@`
  - `%20` for space, `%7E` for `~`
- Q: What about `%` in abstract value?
  - A: Encode it too! `%25`
- aka "percent encoding"

URL Encoding

Reserved characters after percent-encoding

!	#	$	%	&	'	(	)	*	+	,	/	:	;	=	?	@	[	]
%21	%23	%24	%25	%26	%27	%28	%29	%2A	%2B	%2C	%2F	%3A	%3B	%3D	%3F	%40	%5B	%5D

Common characters after percent-encoding (ASCII or UTF-8 based)

newline	space	"	%	-	.	<	>	\	^	_	`	{	|	}	~
%0A *or* %0D *or* %0D%0A	%20	%22	%25	%2D	%2E	%3C	%3E	%5C	%5E	%5F	%60	%7B	%7C	%7D	%7E

Value **Mascot "address": brutus@osu.edu**

Encoding **Mascot%20%22address%22%3A%20brutus%40osu.edu**

MIME

- Multipurpose Internet Mail Extensions
  - Originally for email attachments
- Content Type: How to interpret a file
  - File is a blob of bits (encoding)
  - How should we decode this blob into an (abstract) value? Colors, sounds, characters?
  - Recall: *correspondence relation*
- Syntax: type/subtype
  - text/plain, text/html, text/css, text/javascript
  - image/gif, image/png, image/jpeg
  - video/mpeg, video/quicktime
- Transfer Encoding: How to interpret a msg
  - How to decode the blob of bits that arrived
  - A *layered* encoding
  - Examples: quoted-printable, base64

Example: Multiple Parts


```
MIME-Version: 1.0
Content-Type: multipart/mixed; boundary=aFrontierString

This is a message with multiple parts in MIME format.
--aFrontierString
Content-Type: text/plain

This is the body of the message.
--aFrontierString
Content-Type: application/octet-stream
Content-Transfer-Encoding: base64

PGh0bWw+CiAgPGhlYWQ+CiAgPC9oZWFkPgogIDxib2R5PgogICAgPHA
+VGhpcyBpcyB0aGUg
Ym9keSBvZiB0aGUgbWVzc2FnZS48L3A+CiAgPC9ib2R5Pgo8L2h0bWw
+Cg==
--aFrontierString--
```

Example: Content Type


```
MIME-Version: 1.0
Content-Type: multipart/mixed; boundary=aFrontierString

This is a message with multiple parts in MIME format.
--aFrontierString
Content-Type: text/plain

This is the body of the message.
--aFrontierString
Content-Type: application/octet-stream
Content-Transfer-Encoding: base64

PGh0bWw+CiAgPGhlYWQ+CiAgPC9oZWFkPgogIDxib2R5PgogICAgPHA
+VGhpcyBpcyB0aGUg
Ym9keSBvZiB0aGUgbWVzc2FnZS48L3A+CiAgPC9ib2R5Pgo8L2h0bWw
+Cg==
--aFrontierString--
```

Example: Transfer Encoding


```
MIME-Version: 1.0
Content-Type: multipart/mixed; boundary=aFrontierString

This is a message with multiple parts in MIME format.
--aFrontierString
Content-Type: text/plain

This is the body of the message.
--aFrontierString
Content-Type: application/octet-stream
Content-Transfer-Encoding: base64

PGh0bWw+CiAgPGhlYWQ+CiAgPC9oZWFkPgogIDxib2R5PgogICAgPHA
+VGhpcyBpcyB0aGUg
Ym9keSBvZiB0aGUgbWVzc2FnZS48L3A+CiAgPC9ib2R5Pgo8L2h0bWw
+Cg==
--aFrontierString--
```

Layered Encoding

source
(image)

Content-Type
image/jpeg

content
(bits)

ffd8ffe000104a464946…

Content-Transfer-Encoding
???

transfer encoded
(channel)
ASCII

/9j/4AAQSk…

Encoding (Binary) Data in ASCII

- Binary data: Any byte value is possible
  - 00 to FF (*i.e.* xxxx xxxx)
- ASCII data: bytes start with 0
  - 00 to 7F (*i.e.* 0xxx xxxx)
- Problem: a channel that needs ASCII
  - Encoding must use ASCII alphabet
- Hex: 4 bits becomes 1 ASCII character

  1101 0110 1100 1111 0011 1001

  D 6 A F 2 5
  - Problem?

Quoted-Printable Encoding

- Observation: bytes that happen to be ASCII do not need to be encoded
  - If most data is text, savings are significant
- For each byte:
  - If first bit is 0, do nothing
  - If first bit is 1, encode with 3 bytes: **`=XY`** where XY is the hex value of byte
- Limit line length to 76 characters
- Finish lines with "="
- Q: What if data contains the byte "="?

Example


```
J'interdis aux marchands de vanter trop leur marchandises. Car ils se font vite pédagogues et
t'enseignent comme but ce qui n'est par essence qu'un moyen, et te trompant ainsi sur la route à suivre
les voilà bientôt qui te dégradent, car si leur musique est vulgaire ils te fabriquent pour te la vendre
une âme vulgaire.
```

```
J'interdis aux marchands de vanter trop leur marchandises. Car ils se font =
vite p=C3=A9dagogues et t'enseignent comme but ce qui n'est par essence qu'=
un moyen, et te trompant ainsi sur la route =C3=A0 suivre les voil=C3=A0 bi=
ent=C3=B4t qui te d=C3=A9gradent, car si leur musique est vulgaire ils te f=
abriquent pour te la vendre une =C3=A2me vulgaire.
```

Encoding Binary Data

- What if most data is *not* ASCII?
  - Raw (base 256): 8 bits are a digit (byte)

```
1101 0110 1100 1111 0010 0101
?         ?              %
```

  - Hex (base 16): 4 bits → digit (byte)

```
1101 0110 1100 1111 0011 1001
D    6    A    F    2    5
```

  - Quoted-Printable: 8 bits → 3 bytes

```
1101 0110 1100 1111 0011 1001
=D   6    =A   F    %
```

  - Can we do better?

Encoding Binary Data

- What if most data is *not* ASCII?
  - Raw (base 256): 8 bits are a digit (byte)

```
1101 0110 1100 1111 0010 0101
?         ?         %
```

  - Hex (base 16): 4 bits → digit (byte)

```
1101 0110 1100 1111 0011 1001
D    6    A    F    2    5
```

  - Base 64: 6 bits → 3 digit (byte)

```
1101 0110 1100 1111 0011 1001
1         s         8         5
```

Base64 Alphabet

Value	Char	Value	Char	Value	Char	Value	Char
0	A	16	Q	32	g	48	w
1	B	17	R	33	h	49	x
2	C	18	S	34	i	50	y
3	D	19	T	35	j	51	z
4	E	20	U	36	k	52	0
5	F	21	V	37	l	53	1
6	G	22	W	38	m	54	2
7	H	23	X	39	n	55	3
8	I	24	Y	40	o	56	4
9	J	25	Z	41	p	57	5
10	K	26	a	42	q	58	6
11	L	27	b	43	r	59	7
12	M	28	c	44	s	60	8
13	N	29	d	45	t	61	9
14	O	30	e	46	u	62	+
15	P	31	f	47	v	63	/

en.wikipedia.org/wiki/Base64

Layered Encoding: Base64

source (image)

Content-Type image/jpeg

content (bits)

ffd8ffe000104a464946…

Content-Transfer-Encoding base64

encoded (alphabet)

/9j/4AAQSk…

ASCII

transmission (bits)

2f396a2f34414153 6b…

Base64 Encoding

source ASCII (if <128)	M			a			n		
source octets	77 (0x4d)			97 (0x61)			110 (0x6e)		
Bit pattern	0 1 0 0 1 1 0 1			0 1 1 0 0 0 0 1			0 1 1 0 1 1 1 0		
Index	19			22		5		46	
Base64-encoded	T			W		F		u	
encoded octets	84 (0x54)			87 (0x57)		70 (0x46)		117 (0x75)	

Text content	M																							
ASCII	77 (0x4d)								0 (0x00)								0 (0x00)							
Bit pattern	0	1	0	0	1	1	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Index	19								16				0				0							
Base64-encoded	T								Q				=				=							

en.wikipedia.org/wiki/Base64

Determining MIME Content Type

- The sender (web server) determines MIME (content) type of document being sent
  - Rules map file extensions to MIME types
- If file arrives without MIME info, receiver has to guess (see **`file`** command)
  - File extension *may* help
  - Contents *may* help: magic number at start
    - JPG: **`ff d8`**…
    - PDF: **`25 50 44 46`**… (ie %PDF)
    - PNG: **`89 50 4e 47 0d 0a 1a 0a`**… (ie .PNG…)
- Some types handled by browser itself
- Others require plugin or application
- Experimental MIME subtypes: `x-`
  - application/x-gzip

Summary

- IP address are unique on network
  - IPv4 vs IPv6
- DNS maps strings to IP addresses
  - Domains nested hierarchically
- URLs identify resources on network
  - Scheme, host, path
- MIME type defines a file's encoding
  - Correspondence
  - Layered encodings are possible too