CSE 3231
Computer Networks

More Application Protocols

William Allen, PhD

Spring 2022

File Transfer Protocol (FTP)

- FTP is a TCP application that supports file transfers between two hosts, normally with one acting as the client, initiating the connection, and the other as the server
    - However, once they are connected, files can be transferred in either direction
- FTP does not encrypt connections, thus both the login process and the contents of transferred files are exposed online

File Transfer Protocol

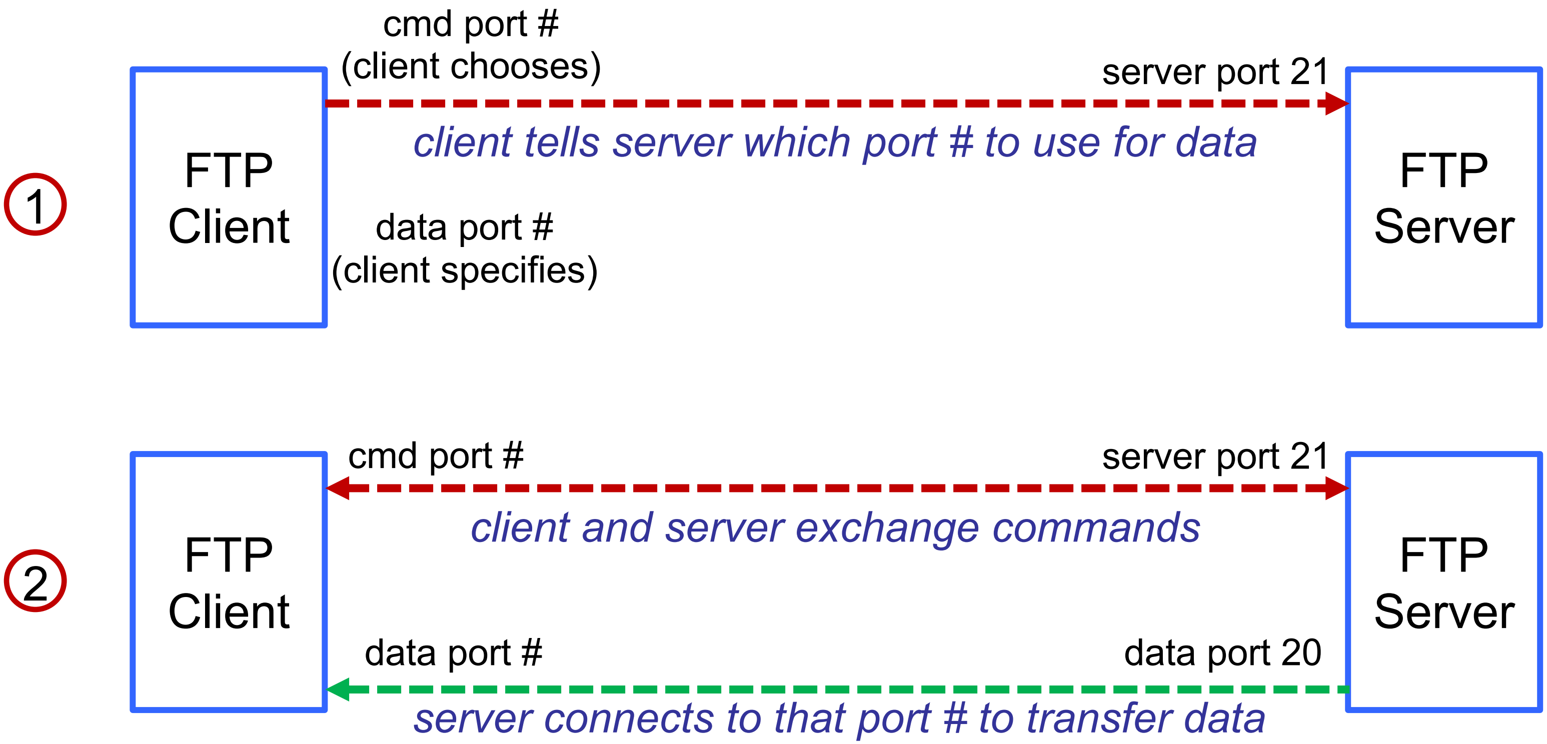
- FTP was originally proposed in 1971 (RFC 114) and actually predates TCP/IP
    - It was first updated to work with TCP in 1980 (RFC 765) and revised in later versions
- FTP is unusual in that it uses two different socket connections between hosts
    - one, using *port 21*, exchanges commands
    - the other, using *port 20*, transfers the data
    - This allows the client and server to exchange “out of band” messages during data transfer

Transfer Modes

- FTP can run in *active* or *passive* modes
  - in active mode, the client connects to the server on the server’s port 21 and waits for the server to establish a data transfer connection from the server’s port 20
  - passive mode is used when the *server cannot initiate* the data transfer connection because the client is behind a firewall that blocks incoming connections or NAT is used
    - the server sends the port number for data to the client and allows it to create the data connection

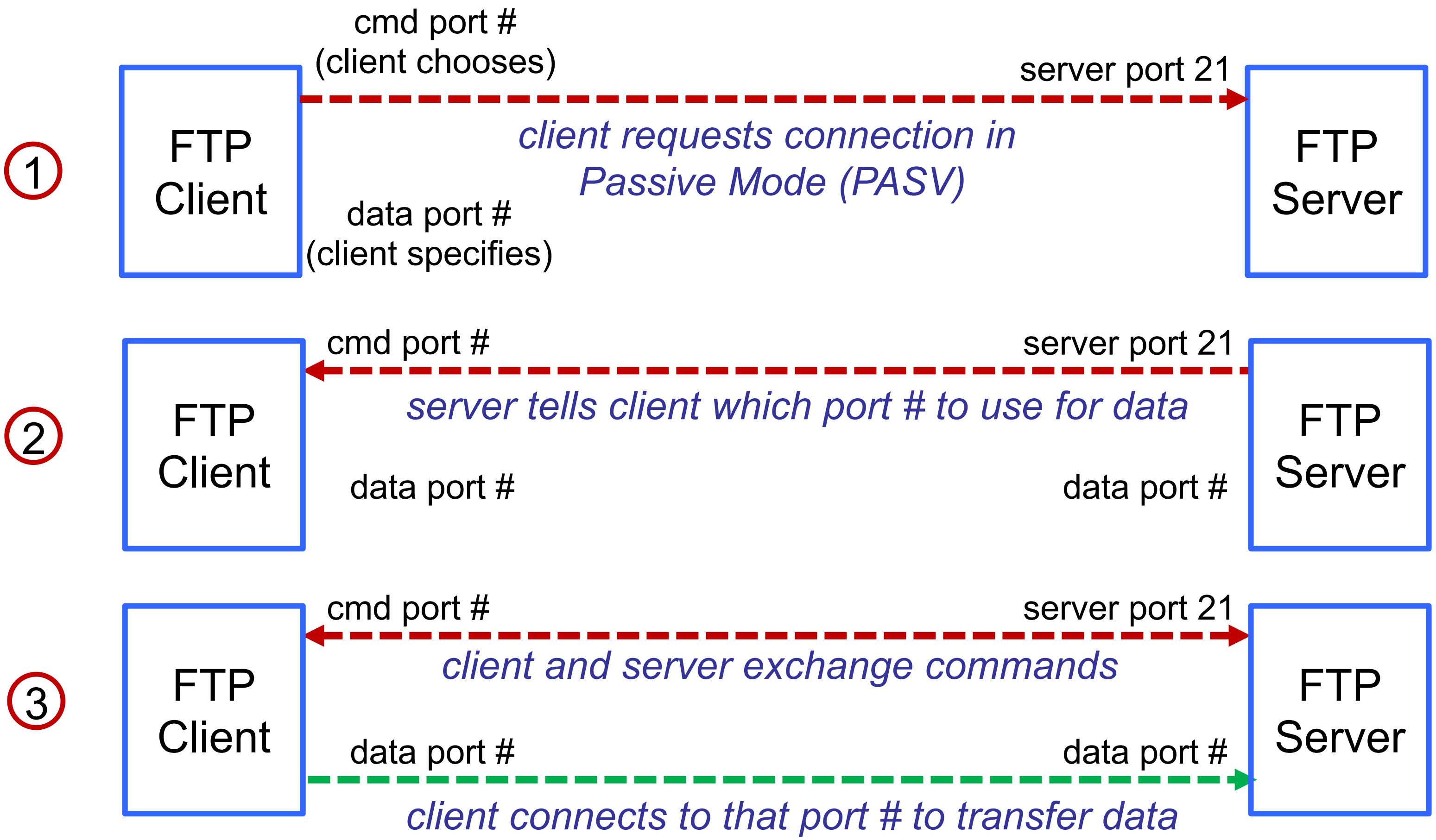
FTP Active Mode

- If there are no incoming connection restrictions at the client end, active mode can be used



FTP Passive Mode

- The client can request passive mode by sending the PASV command to the server



Data Transfer Modes

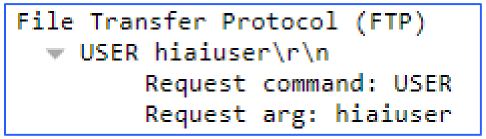
- FTP can exchange binary data or ASCII text and, if in ASCII mode, can convert CRLF between Windows and Linux
- Data can be transferred in several ways, including stream, block or compressed
  - However, block and compressed will require extra processing time for the FTP software
  - Many web browsers included support for FTP file transfers, but this feature has now been removed due to privacy concerns

Anonymous FTP

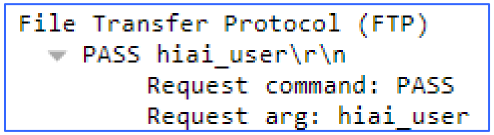
- In cases where available files were public, there was no need for users to login and the *anonymous* mode was provided to allow anyone to download the files
  - users simply entered "`anonymous`" at the login prompt
  - some servers asked for users to enter their email address as the password, but it was not used for authentication, just to track the number of connections by that user

FTP Commands

- FTP exchanges command and status codes over the connection to port 21
  - the client sends the user's login ID with the command "USER" and the user's password with "PASS"



```
File Transfer Protocol (FTP)
  ▾ USER hiaiuser\r\n
        Request command: USER
        Request arg: hiaiuser
```



```
File Transfer Protocol (FTP)
  ▾ PASS hiai_user\r\n
        Request command: PASS
        Request arg: hiai_user
```

  - files are uploaded with the "STOR" command and downloaded with the "RETR" command
  - commands can also change the current directory, list files, close the connection, etc.

Secure FTP (SFTP)

- *Secure FTP* (also called SSH FTP) is designed to replace FTP for file transfers
  - Several other options for secure data transfer exist, including Secure Copy (SCP) and running the rsync or ftp file transfer programs through an SSH tunnel
- The earliest versions of SFTP were published in 2001 and it has been revised several times
- File servers can either run an SFTP server or use an SSH server that supports file transfers
  - one of the most common methods is to run the OpenSSH server which also supports SFTP, SCP

Multipurpose Internet Mail Extensions

- Multipurpose Internet Mail Extensions (MIME) was created to allow non-text data to be included in text-only email
  - MIME is specified in RFC 2045-2047 and RFC 2088-2089, etc.
- MIME is also used in other text-based protocols, such as HTTP
- Email can include multiple MIME sections each containing a different media type

S/MIME (Secure MIME)

- MIME has been extended to provide for the encryption and signing of documents
- S/MIME can provide:
  - client and server authentication
  - message integrity using message digests
  - data privacy and security through encryption
  - non-repudiation (using digital signatures)

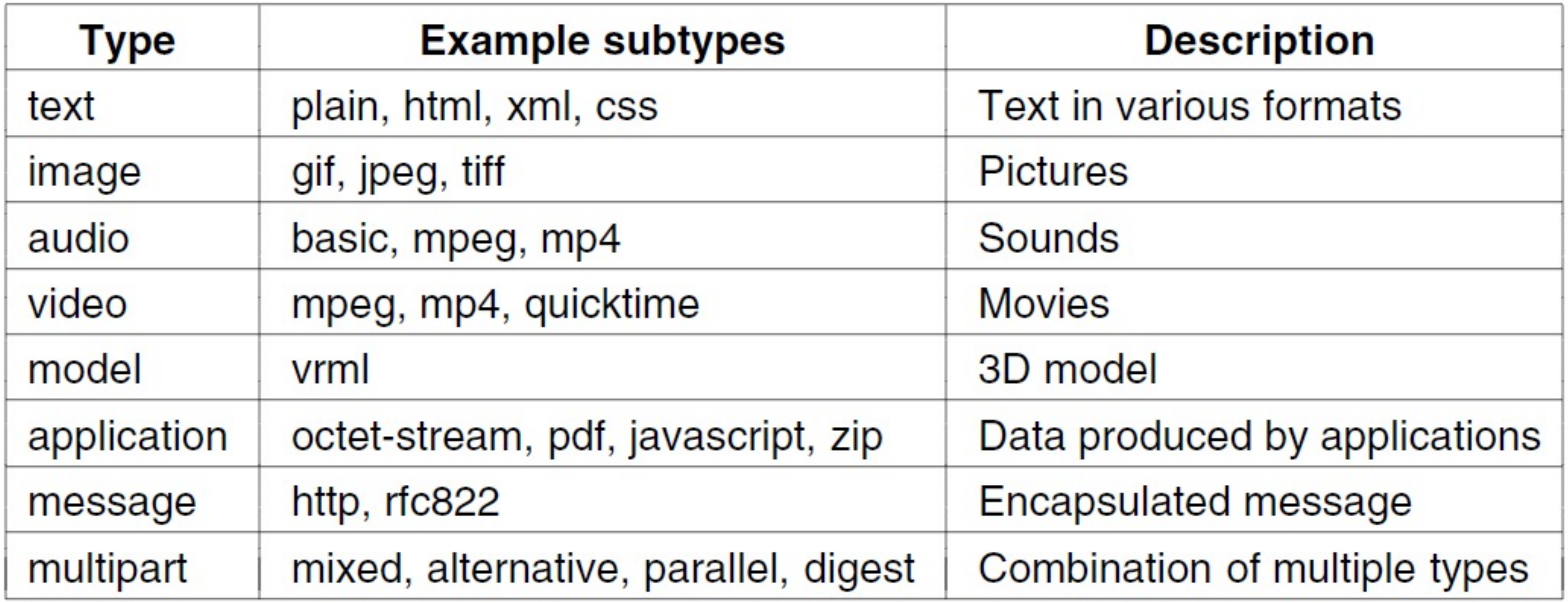
MIME

MIME consists of three basic components:

- **First**: a set of header lines that extend RFC 822
  - *MIME-Version*: (the version of MIME being used)
  - *Content-Description*: (what’s in the message)
  - *Content-Type*: (the type of data contained in the message)
  - *Content-Id:* (a unique identifier for the content)
  - *Content-Transfer-Encoding* (how the data is encoded)
- **Second**: definitions for a set of *content types*
  - For example, MIME defines two different image types: image/gif and image/jpeg
- **Third**: a way to *encode* the various data types so they can be included in an ASCII email message
  - Converting binary data into ASCII text

MIME

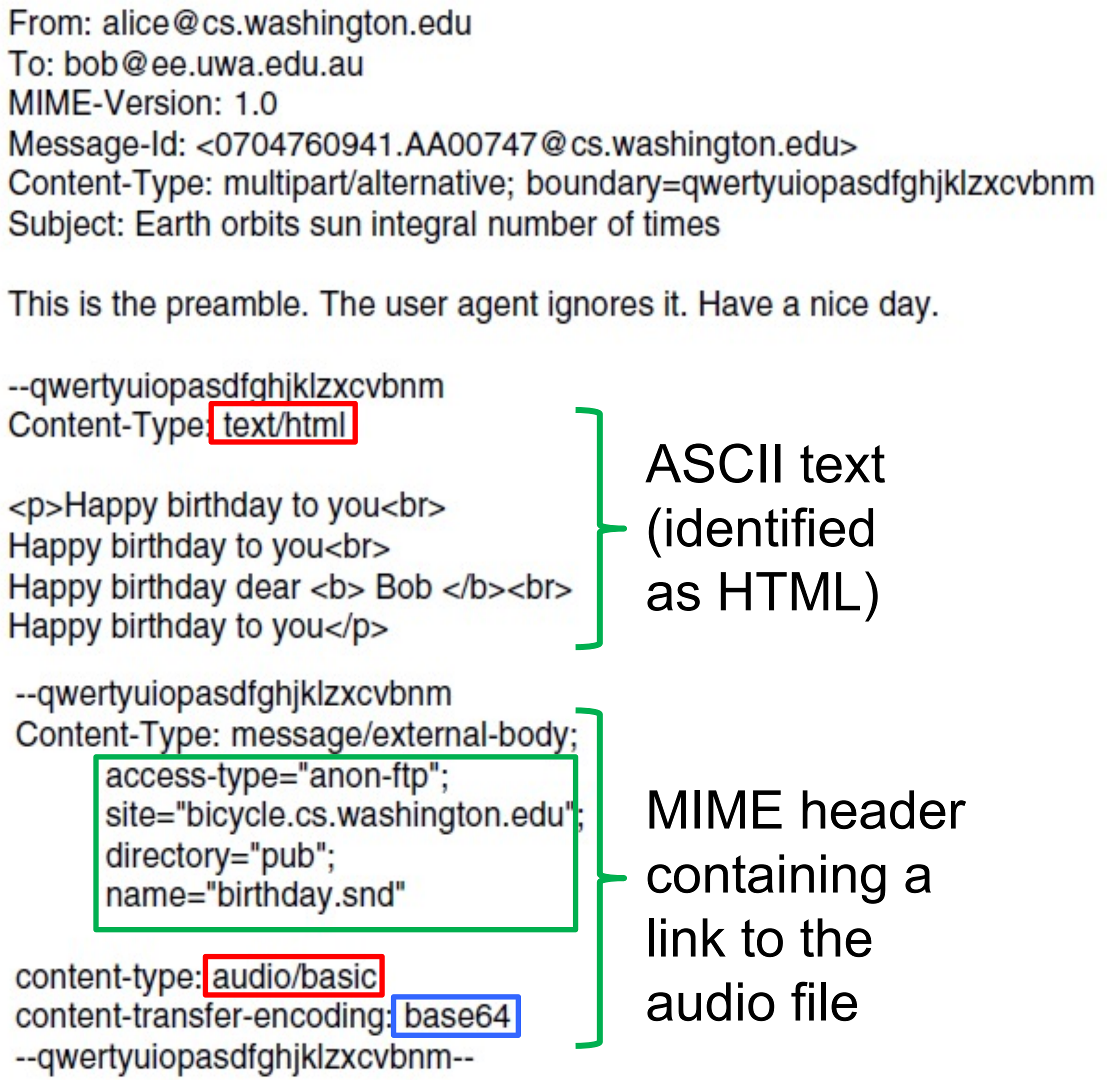
MIME supports a variety of common media and data types



Type	Example subtypes	Description
text	plain, html, xml, css	Text in various formats
image	gif, jpeg, tiff	Pictures
audio	basic, mpeg, mp4	Sounds
video	mpeg, mp4, quicktime	Movies
model	vrml	3D model
application	octet-stream, pdf, javascript, zip	Data produced by applications
message	http, rfc822	Encapsulated message
multipart	mixed, alternative, parallel, digest	Combination of multiple types

An Email with MIME Data

Putting it all together: a multipart message containing both HTML and audio



```
From: alice@cs.washington.edu
To: bob@ee.uwa.edu.au
MIME-Version: 1.0
Message-Id: <0704760941.AA00747@cs.washington.edu>
Content-Type: multipart/alternative; boundary=qwertyuiopasdfghjklzxcvbnm
Subject: Earth orbits sun integral number of times

This is the preamble. The user agent ignores it. Have a nice day.

--qwertyuiopasdfghjklzxcvbnm
Content-Type: text/html

<p>Happy birthday to you<br>
Happy birthday to you<br>
Happy birthday dear <b> Bob </b><br>
Happy birthday to you</p>
 --qwertyuiopasdfghjklzxcvbnm
 Content-Type: message/external-body;
         access-type="anon-ftp";
         site="bicycle.cs.washington.edu";
         directory="pub";
         name="birthday.snd"

 content-type: audio/basic
 content-transfer-encoding: base64
 --qwertyuiopasdfghjklzxcvbnm--
```

Part 1 (HTML)

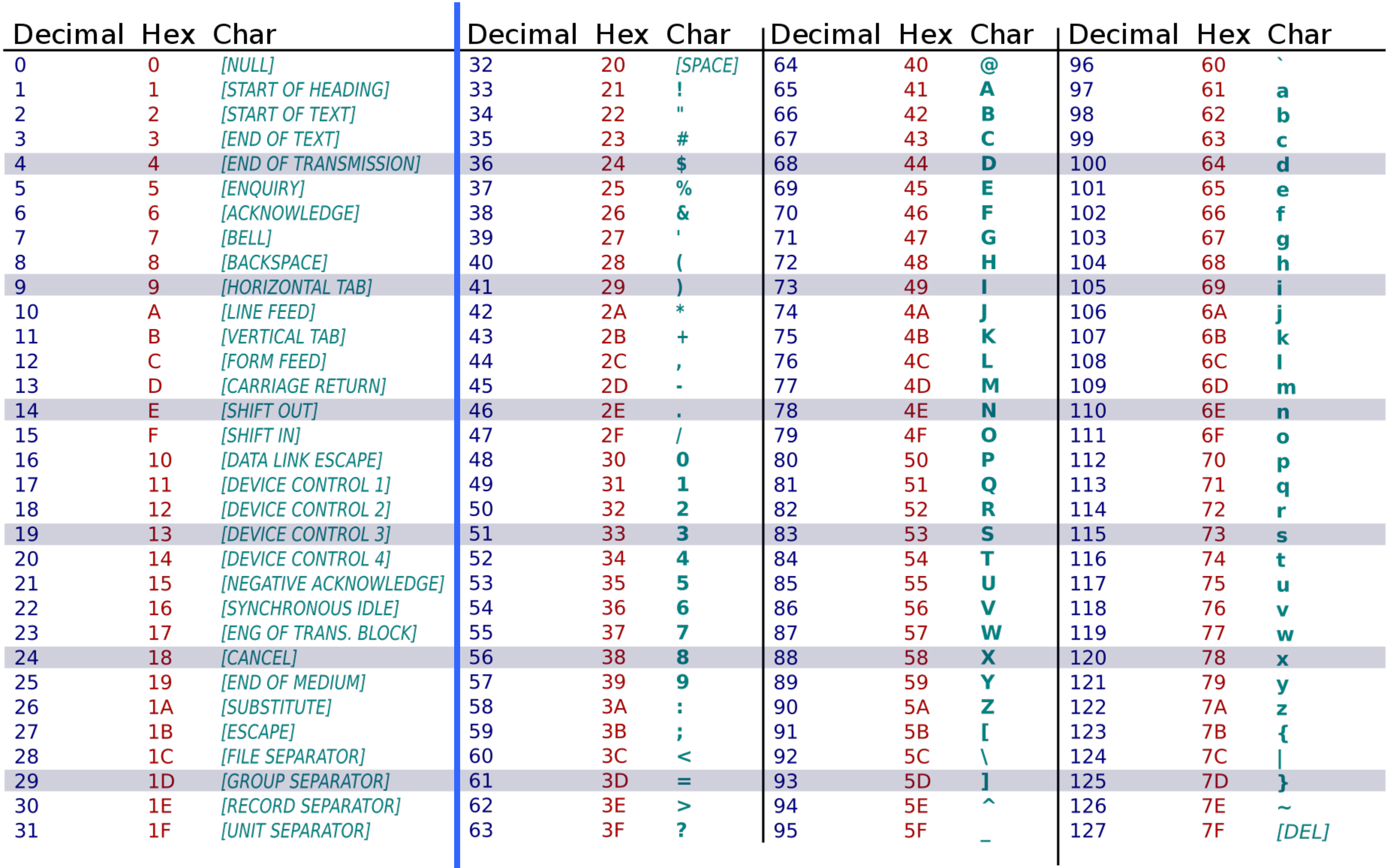
ASCII text (identified as HTML)

Part 2 (audio)

MIME header containing a link to the audio file

Including Non-Text Data in Email

- Since SMTP expects ASCII text in email messages, binary data must be encoded into a format that “appears” to be text
  - ASCII text is 7-bit, using binary values from 0 to 127, some of those values are used as control codes for communication and format
  - Since binary data is 8-bit, the ASCII range of 7-bit values cannot cover all byte values and special encoding techniques are needed
    - one common encoding method is *Base64*



Decimal	Hex	Char	Decimal	Hex	Char	Decimal	Hex	Char	Decimal	Hex	Char
0	0	[NULL]	32	20	[SPACE]	64	40	@	96	60	`
1	1	[START OF HEADING]	33	21	!	65	41	A	97	61	a
2	2	[START OF TEXT]	34	22	"	66	42	B	98	62	b
3	3	[END OF TEXT]	35	23	#	67	43	C	99	63	c
4	4	[END OF TRANSMISSION]	36	24	$	68	44	D	100	64	d
5	5	[ENQUIRY]	37	25	%	69	45	E	101	65	e
6	6	[ACKNOWLEDGE]	38	26	&	70	46	F	102	66	f
7	7	[BELL]	39	27	'	71	47	G	103	67	g
8	8	[BACKSPACE]	40	28	(	72	48	H	104	68	h
9	9	[HORIZONTAL TAB]	41	29	)	73	49	I	105	69	i
10	A	[LINE FEED]	42	2A	*	74	4A	J	106	6A	j
11	B	[VERTICAL TAB]	43	2B	+	75	4B	K	107	6B	k
12	C	[FORM FEED]	44	2C	,	76	4C	L	108	6C	l
13	D	[CARRIAGE RETURN]	45	2D	-	77	4D	M	109	6D	m
14	E	[SHIFT OUT]	46	2E	.	78	4E	N	110	6E	n
15	F	[SHIFT IN]	47	2F	/	79	4F	O	111	6F	o
16	10	[DATA LINK ESCAPE]	48	30	0	80	50	P	112	70	p
17	11	[DEVICE CONTROL 1]	49	31	1	81	51	Q	113	71	q
18	12	[DEVICE CONTROL 2]	50	32	2	82	52	R	114	72	r
19	13	[DEVICE CONTROL 3]	51	33	3	83	53	S	115	73	s
20	14	[DEVICE CONTROL 4]	52	34	4	84	54	T	116	74	t
21	15	[NEGATIVE ACKNOWLEDGE]	53	35	5	85	55	U	117	75	u
22	16	[SYNCHRONOUS IDLE]	54	36	6	86	56	V	118	76	v
23	17	[ENG OF TRANS. BLOCK]	55	37	7	87	57	W	119	77	w
24	18	[CANCEL]	56	38	8	88	58	X	120	78	x
25	19	[END OF MEDIUM]	57	39	9	89	59	Y	121	79	y
26	1A	[SUBSTITUTE]	58	3A	:	90	5A	Z	122	7A	z
27	1B	[ESCAPE]	59	3B	;	91	5B	[	123	7B	{
28	1C	[FILE SEPARATOR]	60	3C	<	92	5C	\	124	7C	|
29	1D	[GROUP SEPARATOR]	61	3D	=	93	5D	]	125	7D	}
30	1E	[RECORD SEPARATOR]	62	3E	>	94	5E	^	126	7E	~
31	1F	[UNIT SEPARATOR]	63	3F	?	95	5F	_	127	7F	[DEL]

← Control Codes → **← Codes used for printable characters →**

Base64 Encoding

- Base64 maps 64 different bit patterns onto a combination of ASCII characters, digits and punctuation
- However, it only uses 6 bits to represent the 64 different patterns and the 8-bit bytes of binary data must be re-grouped into 6-bit blocks before remapping

For example:

three bytes of data: `01010101 11001100 10011001`

is re-grouped into: `010101 011100 110010 011001`

Base64 - Binary to Character Mapping

Index	Binary	Char	Index	Binary	Char	Index	Binary	Char	Index	Binary	Char
0	000000	A	16	010000	Q	32	100000	g	48	110000	w
1	000001	B	17	010001	R	33	100001	h	49	110001	x
2	000010	C	18	010010	S	34	100010	i	50	110010	y
3	000011	D	19	010011	T	35	100011	j	51	110011	z
4	000100	E	20	010100	U	36	100100	k	52	110100	0
5	000101	F	21	010101	V	37	100101	l	53	110101	1
6	000110	G	22	010110	W	38	100110	m	54	110110	2
7	000111	H	23	010111	X	39	100111	n	55	110111	3
8	001000	I	24	011000	Y	40	101000	o	56	111000	4
9	001001	J	25	011001	Z	41	101001	p	57	111001	5
10	001010	K	26	011010	a	42	101010	q	58	111010	6
11	001011	L	27	011011	b	43	101011	r	59	111011	7
12	001100	M	28	011100	c	44	101100	s	60	111100	8
13	001101	N	29	011101	d	45	101101	t	61	111101	9
14	001110	O	30	011110	e	46	101110	u	62	111110	+
15	001111	P	31	011111	f	47	101111	v	63	111111	/

Base64 Encoding

- Then, the 6-bit groups can be mapped into the characters in the Base64 table

For example:

binary data: 01010101 11001100 10011001

grouped into: 010101 011100 110010 011001

This maps into the Base64 encoding as:

010101 = ‘V’

011100 = ‘c’

110010 = ‘y’

011001 = ‘Z’

Represented in the email message as “**VcyZ**”

Base64 Encoding Process

After the 8-bit binary data is converted into 6-bit values: 010101 011100 110010 011001

Those 6-bit values are converted into 7-bit ASCII by adding a leading 0:

```
0010101 = 'V'  0011100 = 'c'
0110010 = 'y'  0011001 = 'Z'
```

And the converted message is transmitted as text in an email or web page

Then, the receiver regroups the bits so that **VcyZ** becomes: 01010101 11001100 10011001 again

Base64 Example

- The following message will be encoded:

  The three rules of the Librarians of Time and Space are: 1) Silence; 2) Books must be returned no later than the date last shown; and 3) Do not interfere with the nature of causality.-- Terry Pratchett, Guards! Guards!

Base64 Example

*VGhlIHRocmVlIHJ1bGVzIG9mIHRoZSB
MaWJyYXJpYW5zIG9mIFRpbWUgYW5k
IFNwYWNlIGFyZTogMSkgU2lsZW5jZTsg
MikgQm9va3MgbXVzdCBiZSByZXR1cm
5lZCBubyBsYXRlciB0aGFuIHRoZSBkYX
RlIGxhc3Qgc2hvd247IGFuZCAzKSBEby
Bub3QgaW50ZXJmZXJlIHdpdGggdGhlIG
5hdHVyZSBvZiBjYXVzYWxpdHkuLS0gV
GVycnkgUHJhdGNoZXR0LCBHdWFyZH
MhIEd1YXJkcyE=*

Base64 Encoding

- Note: the output is larger than the input
  - The input text was 218 8-bit characters (including spaces) = 1,744 bits
    - However, that doesn’t divide equally into a whole number of 6-bit groups (290.666...)
  - It was padded with 0’s to 1,752 bits so it also could be divisible by 6, giving 292 groups
  - The padding is 8 bits (1,752-1,744 = 8) and the last 6-bit group would be `000000`
    - Since that is not a valid character, the symbol ‘=‘ is used to indicate that it is actually padding

Base64 Encoding

- Starting with this image:

- The image is 4,078 bytes = 32,624 bits
- This is 5,437.33 groups of 6 bits, which must be padded to 32,640 bits = 5440 groups of 6 bits, so it is evenly divisible by both 6 and 8
  - That adds 2 bytes, to the end of the file
  - Those 16 zeros of padding are represented by the ‘==‘ at the end of the file

iVBORw0KGgoAAAANSUhEUgAAADcAAABCCAIAAAB8VtljAAAAAXNSR0IArs4c6QAAAARnQU1BAACxjwv8YQUAAAAJcEhZcwAADsMAAA7DAcdvqGQAAA+D
SURBVGhDzVp7dFVVej/7vO4r95nc3CSQh3kqgxMgEoyAgsxMcc3SWqtMHVtbW2uXsxQGdWaAichDLMNqZ635A3XQ5dCxa7pAZ9rSdsmrKOiAQTrU8JQkhIQAC
SQkN7m5r/Pqb599cpOQ3Dwu/tHfOpz7nW/vs/fvfPv7vv2dE8hrr73G/b8Hb/1+TdB13ZK+VnwNtjQMw+/3r169GrIkyYqShFBYWPzss3/9dZG+XZY2m23Dhk2qqlrXo2
G3y6+88gohxLrOFLe14g0NDfX168elCAN7nHIslnz99TcEQbC0mSJzlj6fd//+g9bFGHi8Wd9+6LvLI3/bZRdhbEmSrIaMkDnL1atfsqTxIApCMBjMzs4uKavgeX79+g2wr
tU2fWTIUtN0QZjIPFjlo0ePHjx44GLzBRZDW7duY00ZIEOWx483TGybcDjS2Hj6+vWeWEJjGrivKIpMni4yZNna2mZJaaAoiiWNwKlTpy1pmsiQZVPTVzi7XA5ZIPNq5
nJk1Dgs9TgctsWLFzINQ0PDcUuaJjJkyRCLxde9uv7smbOLF9UxDfghR65bt+bdd3fs3LnzyJHPsrKcrAloa2u1pGkic5YgtG3bz5Yv/6OdO9/735N/YEpN07Zt2/b97z+5
dOnSsrKyNWt+PBgZZE23g8xZapq6cuWLmzdvfuKJFU3NLdCIPIclfv75v4tGY6Wld8ycUbBlyxZ9RIxJkg2PYV1MBxmztDmdrje2bDl06JDX683Jyc7Pz61buDA/Pw/Z8
d5770MCWLVqpWFofr/XuuM2kCHL5csfFAS+ra09FkuEQqH29vburhufHz22e/eHGBMJkufF3R/8K85er8e65zaQIUsky4GB/l/u2AHvzBXamnfNvvxf8vnfSttXO9r2iJq
m8ARVnApbXrt2zbrnNpDxihuClJ5qbNyzqfidH/PV1RwROE+++L2niwNlFX1fursOyAf/Ze35803J5LAjwildruGQnzoyZEnMBHnu1/flOTrcAYJhVJFwgs/uyBXFkGyXHT
NdddVvzRx4rDB4S0GUSRWXIctly5bmZtv9bs7lJi5ZMHhOshFetBMuC1sMI4zpidgHH0Y6v2pVjVHbfTQataTpIGNbkr7+ZBgbTBbH8xzoEVkQBS/hjmkJpbVF/Oqwui
BP0KOkszuG/gg1diMyFxOmhQxZAomkvudzz1u/hYMagsstO0I817Tvo8hv3komO+IOOzcQsZ8x/oY3S+AHH3wQGxK2JUS9xzPtqM+Qpa7TiuiZH73/zaXP7vlP5WZ
3/5WzbU0fx+4QSW0VH+mX2I1r+MX7c2d/78kn/wz9P/nkExTC7O3i5ZdfZoNMHRmyPHz4CKLgZm9fbuUjUtUPDv1Oaz/D94WNcy3GtR5NuW8/CSx0utzz5s0FJzBT
FHXDho14u0AKy8CcmbydYSbMB6G1teXGjRsD/WGDiD2XjvLJzkDlHxtqXNVJReWdxcWFrP/82gUn//AlkxlkgfvJup9aF1PAtG0Jilu3boUha+Z+s7BwRmVIBS2HdSW
7aL6//GFD1/05+fNq5peUFFk3cNyJL06YDjIM5NC+PsTeVDFtltu3v6koGmY9fuIELrOyXMFQCNwhG4Zededds2bdFQj4KPUhYAeqq62xLoaAcUb2mRjTY9nd3dPb2
wc/e//9fwInpvzGrLtq5tfOq7In4aL78/NDkjTOa8PRzxssaQjY6zdt2jRFotNgabPJb7/9SwiY4NFHH2VKBofD7vFkyXLa15pkMmEb04ow2rXrA+tiQkyVJbbg9es3skd//PH
HbrGBKAqJBP3wMhYwfG9veGAgEovHLdUINDU17d49OVFhyZIljghzD3D2oJffPEFsPT5/CglYVfUvF1dXYOD0WRSsdvtOKO1p+cmjq6u62gCwuH+7OzA739/lI0w
Ej09PYsXL8LDWNfjYUq2xMs/VofJdlnEfBCSySTYyLLkdrtQhBcXFwWD2dgJscHANSE7HA50C4cHVFWLxaJPPfVUbm4ONCtW/Ck35NMMLK9NgCmxXLlyJf0xEN
mG1++bPfvu2tr5Pp8HFKGhSq8nFUywCmjhANHKyvK6uto5c6rxnD3d3RcuXEDn3bt/5/P7gkH6qAwTGxKYnCXGFUUZK2t3oKYgn376GeaGnn0CgB5K0IXM5oIGb
goNlh4HNLDurFlVFZWVjY2N+/Z9hCH7+vpv3uwTuGGLTmzOyVnKMqVYV7cgSaq0tLSqhlM2QIC
+9/wJyBAj1opzSYnCViEHfDDJAHolFGC5yQmNi4lIEDnNAEitDYbJYnmOGFVmpv6PPycgsLZ8ITAoGA2+3+7MiRUCgIPcPevfstaQwmYQlLnD//1Z1VFXgLEwSC+
ViliFlhI9YHYQSKWFwwgR4PAH44oxeUaGLui54YDZo9e/7t3NmzDQ1f6By5557hPenkyZOs21hMwhKZRddUAQQJOX36FKbEumMs0TQc6wMPg8YEc2JaUOIM0s
xfoYRFTSX1gZqaGt2w7v3s8GHCWczwDOk+c07Csr6+XtX006fPgFRFRTl8ERsMGDNC6lBzIhGDJkWInQHmGxBwF1YAgrn0UFreDNBXtxHfmFIl/S0YX8sQj8eRJ
u+qKud5AUnRXGLLBpgPfJjDmTlIB5ld4kxpmiYEYCFFUcDM4q2roGkOw/n9owrNN99825JGIy1L2H/btn+AYPB0BdeuXQsNwgIFESNBW3heVVWcaRybbhNATl1hrU
6S5CQ2wEHvnRAdokoqqD32Hqay8k93F0NFx2ZJGIy3LvXv3gg2EhLn/Llq0CNNjRDGwmGFk86B5hynBxowYGvKgBw2oo9V0NctZMR4Os8liGYlEmDAx0rl8YZaP
8Lbq6mrwKyouhknAwPQwmgtNCxEElizbGFEcpsYak4UXmtCf0WLPZoL9EDbLpEjLkjUhcRw4cEBVkzZZZZgxgsJGOiDOshV8ocIkmaHAGWCsEdGYyuwvPLAi4hZ
SXI5i8h3egdJiApW63S6se42OxpNft6uy0PvfAYKn8x1Ij8iUIlx5gMMwNjekbrDvlxFimVhkPC8KlpSXNzRdx6bRJ2NjmVN8NuaCgwOxyK9KyfH9d6dUPba8/L/futQ0MJ
JYtW4bxMTfbGFlKZ94G4BURrbBaymZASmB6CFoiHD77q2v/s6O0rKSlpdVcDBJLaseONcg2W1VVxXPPPWeOdyvGZ2lwwtzCPkWjKRBMBj6xd3VFQnl52KthG3Ci
ww+bjW6PUIK0SRdGZXFNQUcw9SJv3NxXJHM/8rleudRMv8qmIBAD0Z2VbFpQ9qmlGo1hloTTFM6pcFkJzrvMttntizlsQ990NK5jL9/T0+vz+q50XMHcmBjk0GJSpA
/DNsmUddEHXNHKHonqCYnHdK5LDXcZvGRj3Rg0g3R1du7b5airatA0a4SRsFjm6l/8Segfnynf+HRh/Z+Hflp0B9K4QARqCQa/xP33z+REQikrr3rjjb83w4MuJRwgF
MoFG1wzipCxf+IMO4IfnBjuiwc4f74JrQnVUDq1+SWj/s6CrLBvg6iGdT7MicY4Lx6MpTHbfTRvhisUIPkFtsISIRgUbbTQHgbsNvcB3kVNYGzcuBnZ2CRBX9lAhRksFS
gwJWhBwFPhSbD6jY2n8Oaga4aW5GKDxtoV4uMLeI+drsa3qsl/rJOKinlB4XiD40UryEbCsqVbjtvEmGAnol0XbXT6W4H7+7kTOyW2t7W2tkmSHbRAhZmNBTiAVlA
Eewiof2FlNKHKQNNgxKGphstD8nONFx/ld62RDm4StzzNl1VyuUHBjuwkcKtr3+LJrUQZSxJPCpEoXBPpy1SMAWbH+gdKxGeWWQ8GXVlZBask4JrMUxnLlaPS1Y
ctQdHpdMKN+8JxTTMcTj6Qw+fmcjNDXG6lCxXw2X4+9UdNPNATFVusiyFYUya0uEq/WIy14Qhgy4noG38ipf4DwaVL7aAIKqyahAZyKoZACGfwR2vQb6M1AFZc5
RBH2T7B7SG+bOL3E6eLR5lFhzOBICwIBP521saRFrVYqqrDXK00lmQgnMQRl5NceJeSgMGWLn2AGS9lSJgWtCBDiQegd5kCEeT771+moXIROWycvMx5fYI/IPh
8ggMzj/6eIMkJf07WS4t/7uStXd4ayC5EknF40IQ8zSZj0AjMsW38qyJFSezfTysStqwAUo/pjIQDIOrN2+CmajBU4hBQ7cPYHI/FNW1HnyX94vGE/8s5/8wTOsiQkxGb
ZKf1qGkFinS7q2SQZFh/4uHOWDwBiuBnxhBNOmYsoQslnQqs7dvfdLncEPIDPJYbhjS7mGOlB2hE+iVtoAsj4ZL+0zkZxYzHI4BZqnAeUUGPBs+5NM7tIP/+ahAxib
VGCJt+SSMM/NjGg9yUiMN40qpVL4EiWvP9HNaKxz6T3n4MepLrDwvXLyvvNL3KNJTLLO1XwRw4MA0zffz/zDIK6BoJ626X+o1C6aGHvhuNRiUJb8OgIqC2D4fDz/
/gBVm2u7K8qS8iQJZMBAnPgBLf0oyEodHVw+xd15TL7eS9L1/4sPtleAdrpT85QvNgRO/tTcQSRlKxqwnRQJnLxhr73Cg4kFwQqoSrLeMPHDjo8+WIIqoFJ844QqG
Cd3a8i+rM6j+EoEtXFbqUtCQaCcKpCT4aIz09avtF4b1zL+2++kNldEDRr1mtZLm77+DNXqGv34j3KrGYNhAxEkldUbCCDp5DpUgNMOwMCum6oXd06N19xrHm8b
aA8TC3mJTncx4X53BIdOfGXbC+avQP8DevGz2d6qHPueNktZk5bx2RsuQ59aq4uEP8zpX+/OuXTsUG9evdMC3f1an39SajYR2Fr6LyNhHvVXxS4W/cUK9dMVra9
Jbr5HSHmWhobaHA/mMP8y2Z5tQZ2fzMgGGTCPUOPL5mxOJ8OKz3dOotl/WPtfqopy5dNIz39d/QiM2ndZ8cPP2bGXmSy068LsOdxTucXMCFpdG6e43mNv1iNzn
ZQRY98kNZppthuv/aZFaihtfr/ejX9Q/frZXM5LO8ot8FY3LxKNcfJVdvcFcL6nUUZOkx2d8oDEN2es7s/0WFpxOp0C4aKCB7ozjI2Sv6khVrx/pfOqjK4MWPf3FvOedxEy
wLVjsSJ1d7CF/96sQUgWn8JSUeHTi39+eOLPtgTOXchTVL/2KqLjkExP2+neuKgqJT5gYTWlS6u3rJIyxPTYxp/70HrkYT8zT5jQRGMH/hzFMdZHxvnQD0/f82KAIYw
TymPAjH/R9XoLt9c95SLgAAAABJRU5ErkJggg==

Other Uses of Base64 Encoding

- In addition to MIME attachments, Base64 is used for embedding binary data in XML documents and binary data in scripts
- Base64 encoded text can avoid issues where characters in a document are incorrectly interpreted as delimiters
- Spammers use Base64 encoded text to avoid detection since the actual contents of the text can’t easily be scanned

Data Compression

- To reduce the bandwidth requirements for data transfer, various data compression techniques can be used
  - Some media protocols use compression by default, such as *jpeg* images, *mpeg* video and *mp3* audio
  - Some VPN software automatically does data compression along with encryption
  - Network Interface Cards (NIC) can also compress/decompress data in hardware

Data Compression

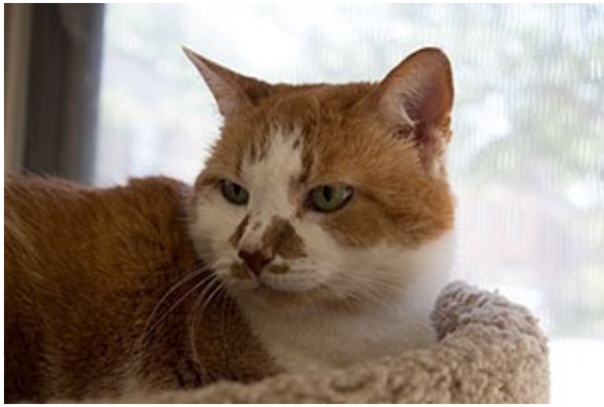
- There are two main types of compression
  - *lossy*, where some less noticeable parts of the data are discarded to reduce its size
    - this is acceptable for media data where the viewer may not notice the removal of small details
    - jpeg, mpeg, mp3, etc.
  - *lossless*, where the compressed version must contain all parts of the original data
    - this is necessary for code, data, executable files, and other sources where complete integrity of the original data is required

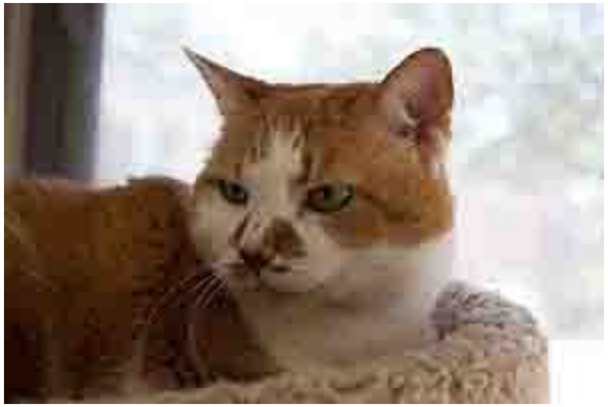
Lossy Data Compression

- Most lossy compression algorithms allow selection of the amount of detail that will be removed, letting users control the balance between lost detail and file size
- When images or video will be shown on youtube or other online media, lossy compression is necessary to control the massive data storage and transmission
    - some sites automatically recompress uploads to meet their storage requirements

Lossy Data Compression

- However, users should avoid losing so much detail that the image or audio quality suffers
  - the images below (from Wikipedia) show the impact of too much image compression
  - both images show compression artifacts, but the image on the right has lost details in the cat's fur and bed, and the compression caused a "ringing" affect





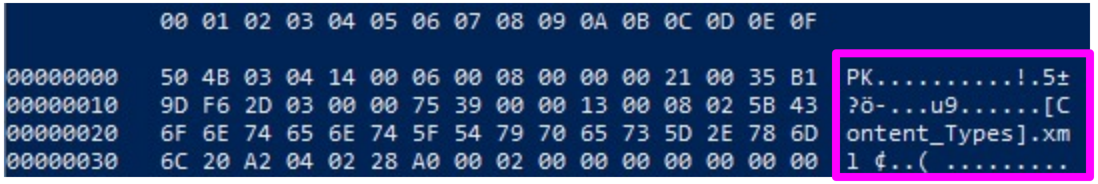
Lossless Data Compression

- Lossless compression is default for some media formats, such as *png* and gif, and is optional for others, like *tiff* and *mpeg4*
- The most common algorithms for data are based on the Lempel-Ziv method, which is used by *7zip*, *gzip*, *rar* and others
  - Lossless compression is particularly useful for text compression since some bit patterns are not used and others are used frequently

File Compression

- Many file formats use compression when the document is stored on disk
  - when the document file is opened by an application, it is decompressed before use
- For example:

  Microsoft Office formats that end in .___x are compressed XML files and have the standard PKZip header in the file



```
          00 01 02 03 04 05 06 07 08 09 0A 0B 0C 0D 0E 0F

00000000  50 4B 03 04 14 00 06 00 08 00 00 00 21 00 35 B1  PK..........!.5±
00000010  9D F6 2D 03 00 00 75 39 00 00 13 00 08 02 5B 43  ?ö-...u9......[C
00000020  6F 6E 74 65 6E 74 5F 54 79 70 65 73 5D 2E 78 6D  ontent_Types].xm
00000030  6C 20 A2 04 02 28 A0 00 02 00 00 00 00 00 00 00  l ¢..( .........
```

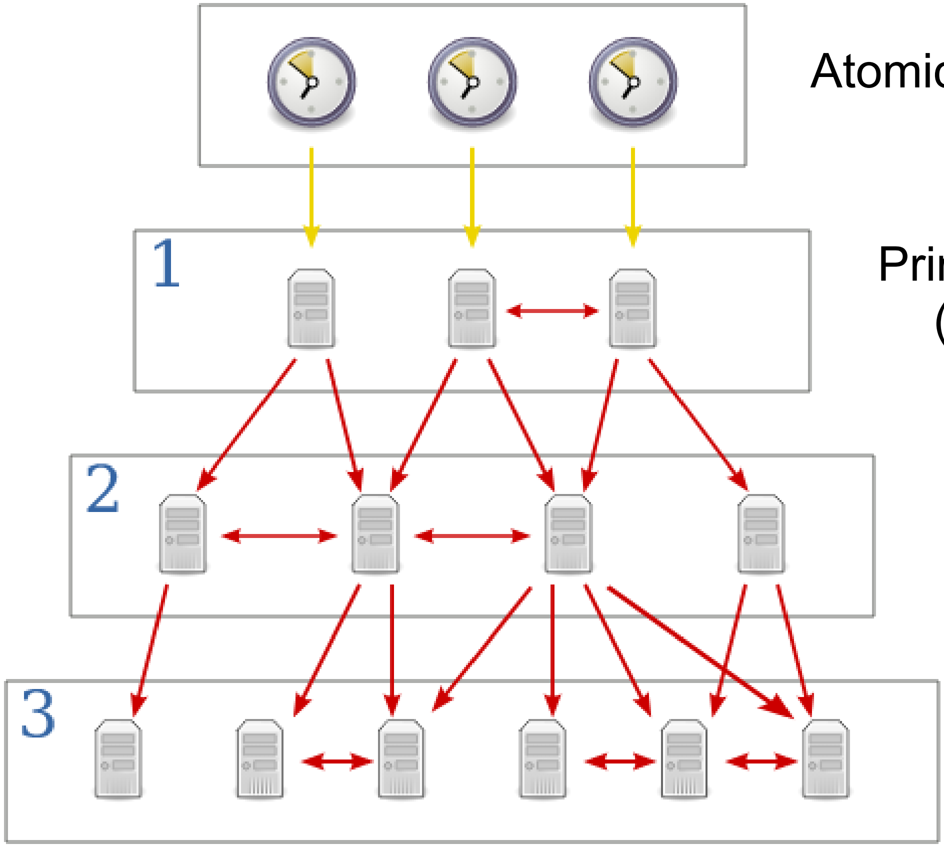
Network Time Protocol (NTP)

- The Network Time Protocol (NTP) has been adopted as a standard for clock synchronization throughout the Internet
  - SNTP (Simple NTP) is compatable with the more complex NTP but easier to implement
- The current version is NTPv4, which is compatable with the widely-used NTPv3, but added IPv6 support and a few other features
  - NTP accepts UDP packets on port `123`
  - NTP servers can be accessed through a “pool” of available servers at “`pool.ntp.org`” instead of connecting directly to a specific server

Network Time Protocol (NTP)

- A number of servers located across the internet are used to implement the protocol and these servers form a synchronization subnet whose levels are called *strata*
  - Servers at *stratum 1* (first level) are called the primary servers. Primary servers have highly accurate clocks (e.g., via a UTC receiver)
  - Servers in *stratum 2* are called secondary servers and are synchronized directly to primary servers
  - Software at the leaf level (*stratum 3*) executes in users' machines within local networks

NTP Architecture



Atomic Clocks

Primary Servers
(stratum 1)

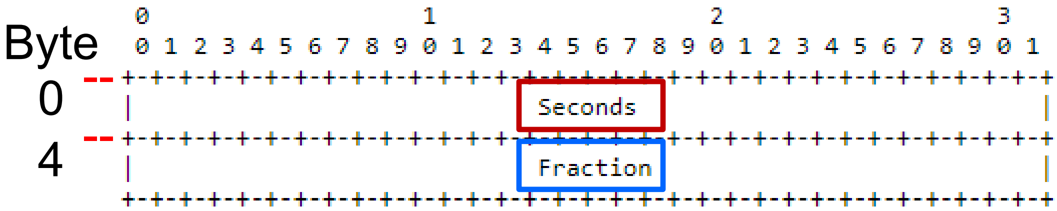
Secondary Servers
(stratum 2)

User’s Hosts
(stratum 3)

NTP Timestamp

The NTP Timestamp format uses two 32-bit numbers to represent time

- The first 32-bits store the unsigned number of seconds since 01/01/1900
    - this value will wrap around in 2036
- The second 32-bits store the number of seconds since midnight, as a fraction
    - this gives sub-microsecond-level accuracy



```
Byte   0                   1                   2                   3
       0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
0   -- +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
       |                           Seconds                             |
4   -- +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
       |                           Fraction                            |
       +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

NTP Header

```
Byte     0                   1                   2                   3
         0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
  0   --+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
        |LI | VN  |Mode |    Stratum     |     Poll      |  Precision   |
  4   --+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
        |                         Root Delay                            |
  8   --+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
        |                         Root Dispersion                       |
 12   --+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
        |                          Reference ID                         |
 16   --+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
        |                                                               |
 20   --+                     Reference Timestamp (64)                  +
        |                                                               |
 24   --+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
        |                                                               |
 28   --+                      Origin Timestamp (64)                    +
        |                                                               |
 32   --+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
        |                                                               |
 36   --+                      Receive Timestamp (64)                   +
        |                                                               |
 40   --+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
        |                                                               |
 44   --+                      Transmit Timestamp (64)                  +
        |                                                               |
 48   --+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
        |                                                               |
        .                                                               .
        .                    Extension Field 1 (variable)               .
        .                                                               .
        |                                                               |
        +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

This diagram represents the header as rows of 32-bit (4 bytes) values

Note the different 64-bit timestamp values contained in the packet

NTP Options and Extensions

There are several timestamp fields:

- origin - when the client packet was sent to the server
- receive - when the server received the client’s packet
- transmit - when the server sent its reply packet
- destination - when the client received the reply

Certain header fields are used to determine how timestamps are used or to provide options

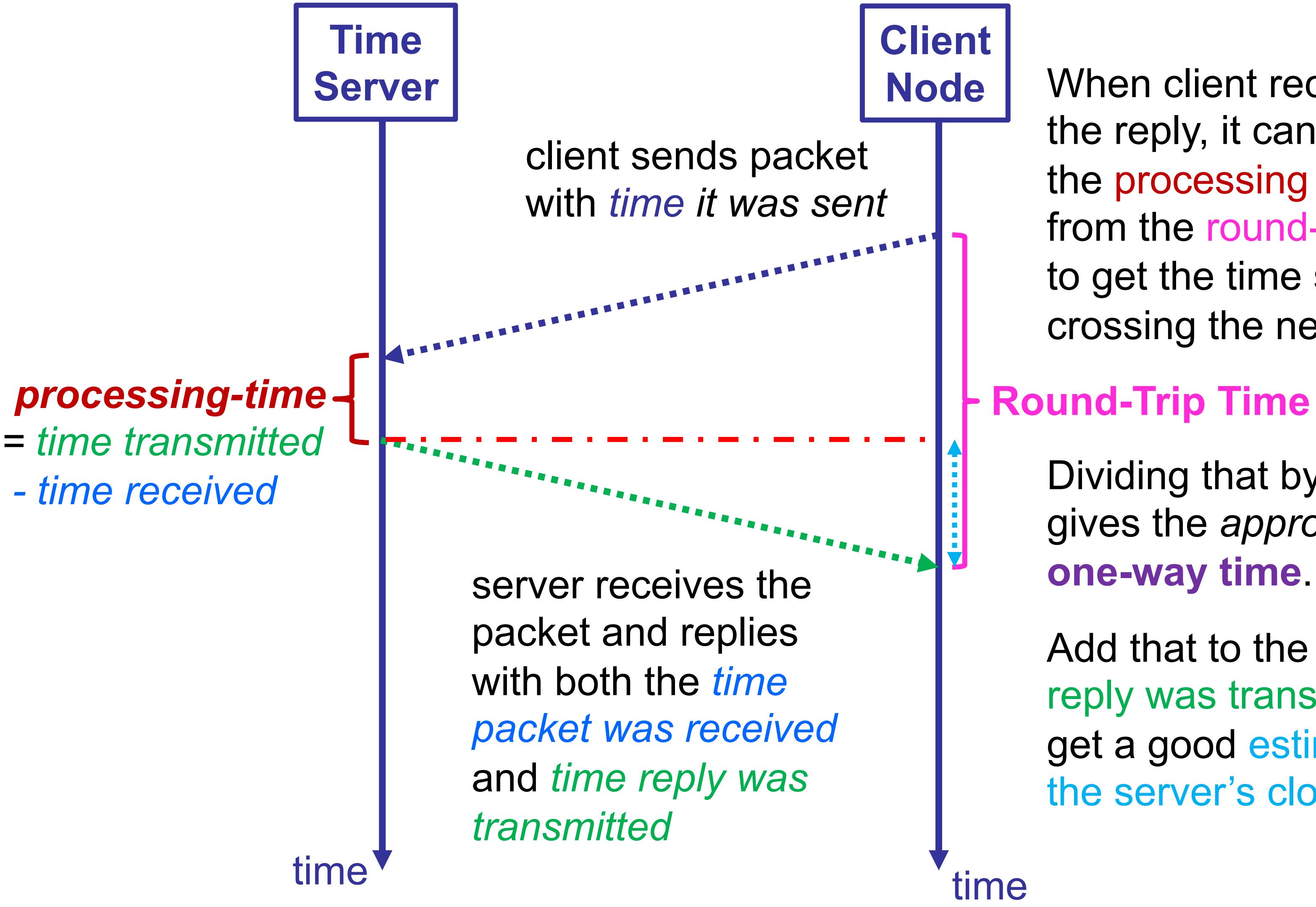
- the date range can be adjusted to deal with the integer wrap-around issue
  - in theory, NTP can represent 1,000’s of years in the past or future by shifting the range of values
- upcoming Leap Seconds can be indicated
- a checksum can be added to detect errors

Accurate Time

NTP uses the exchange of timestamps from packets traveling between the client and server to more accurately determine the time

- The round-trip between client and server gathers four timestamps:
    1. time when a packet left the client
    2. time when a packet arrived at the server
    3. time when a packet left the server
    4. time when a packet arrived at the client
- The client can then average the times to remove the impact of network and processing delays
- Exchanging a series of packets provides an even more accurate calculation

Comparing Local Times



client sends packet with *time it was sent*

processing-time = *time transmitted* *- time received*

server receives the packet and replies with both the *time packet was received* and *time reply was transmitted*

When client receives the reply, it can subtract the processing time from the round-trip time to get the time spent crossing the network

Dividing that by two gives the *approximate* **one-way time**.

Add that to the time the reply was transmitted to get a good estimate of the server’s clock time

T_{sent} = time the client sends the request

$T_{received}$ = time the client receives the reply

$T_{processing} = T_{server\text{-}transmit} - T_{server\text{-}receive}$

$T_{round\text{-}trip} = T_{received} - T_{sent}$

$T_{in\text{-}network} = T_{round\text{-}trip} - T_{processing}$

$T_{one\text{-}way} = (T_{in\text{-}network}) / 2$

Compare client’s clock time with server’s:

Is: $T_{received} == T_{server\text{-}transmit} + T_{one\text{-}way}$?