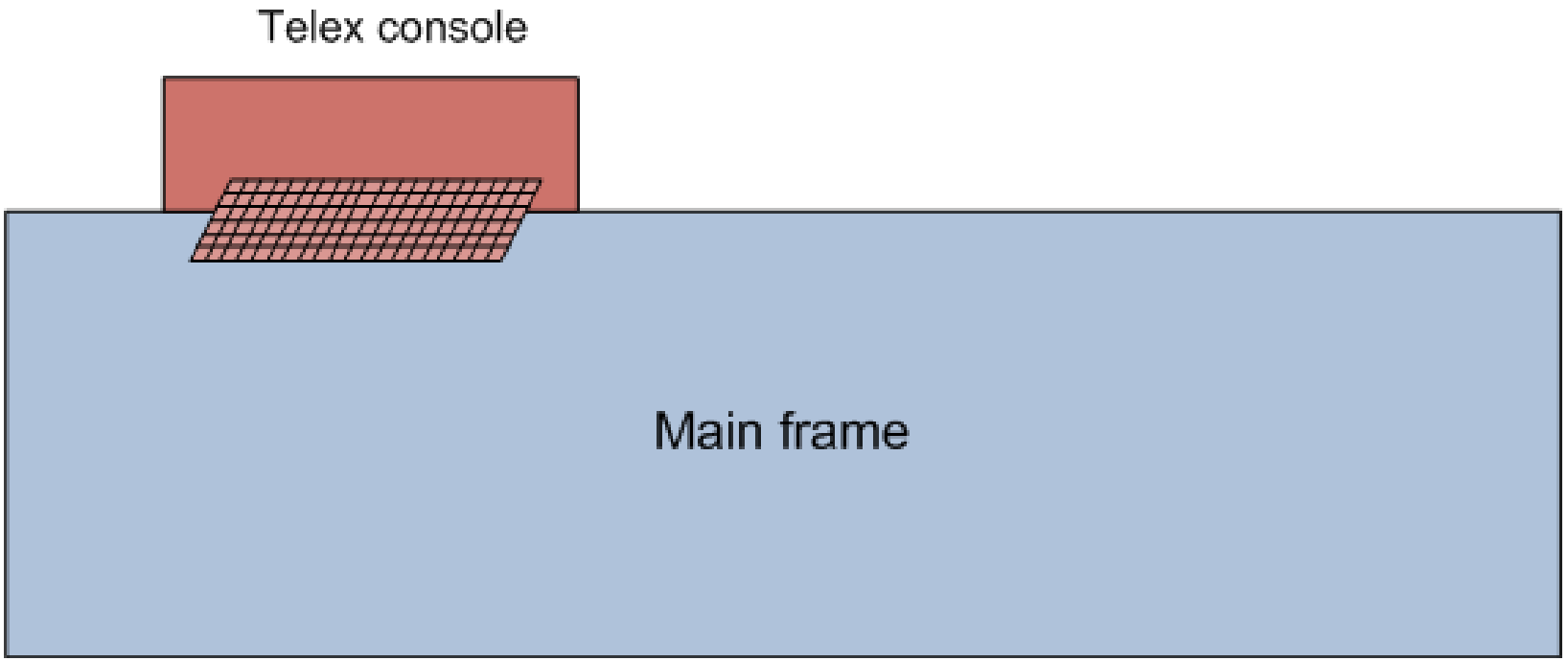
Introduction

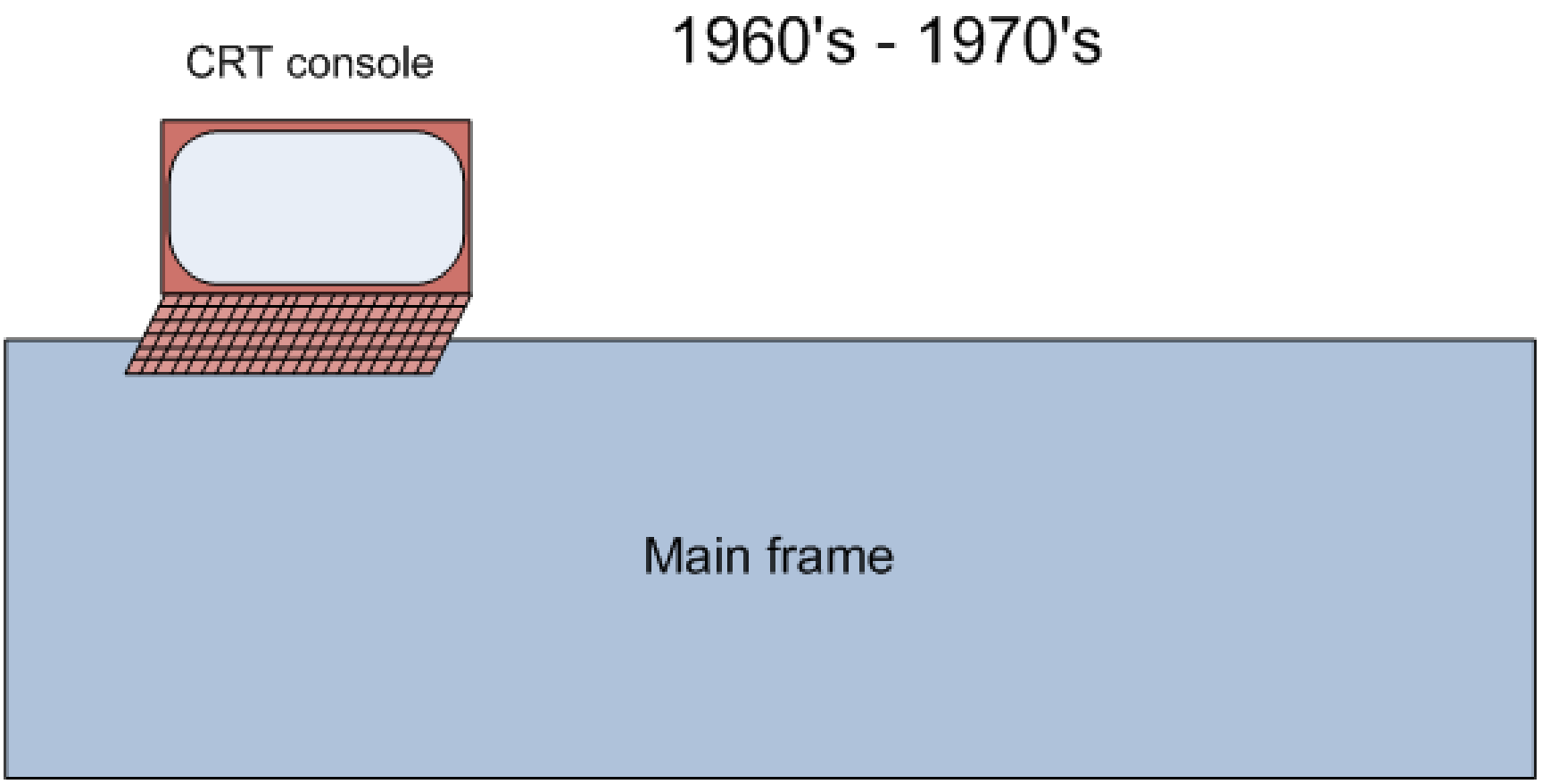
CS 2XA3

Term I, 2018/19

1950's - 1960's

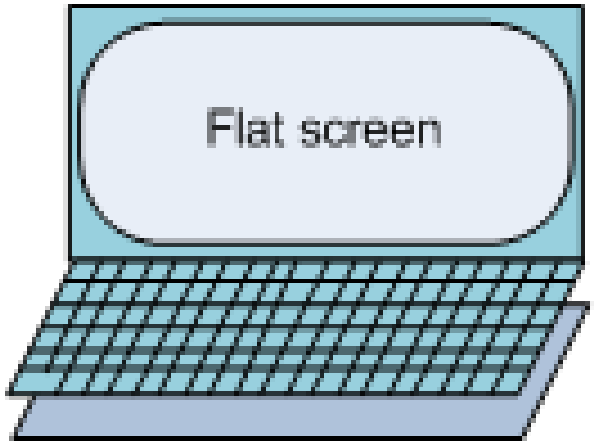


1960's - 1970's

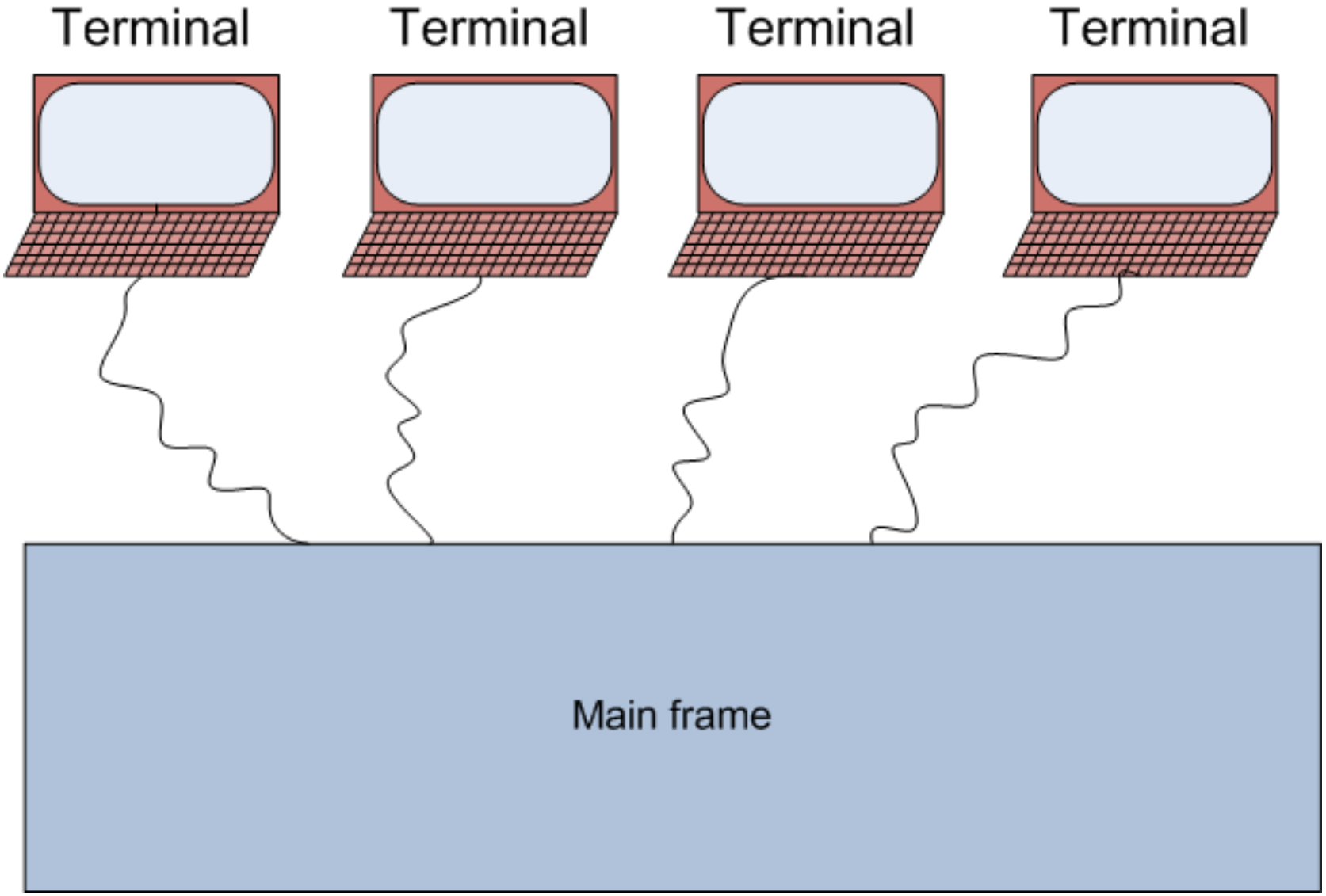


Laptop

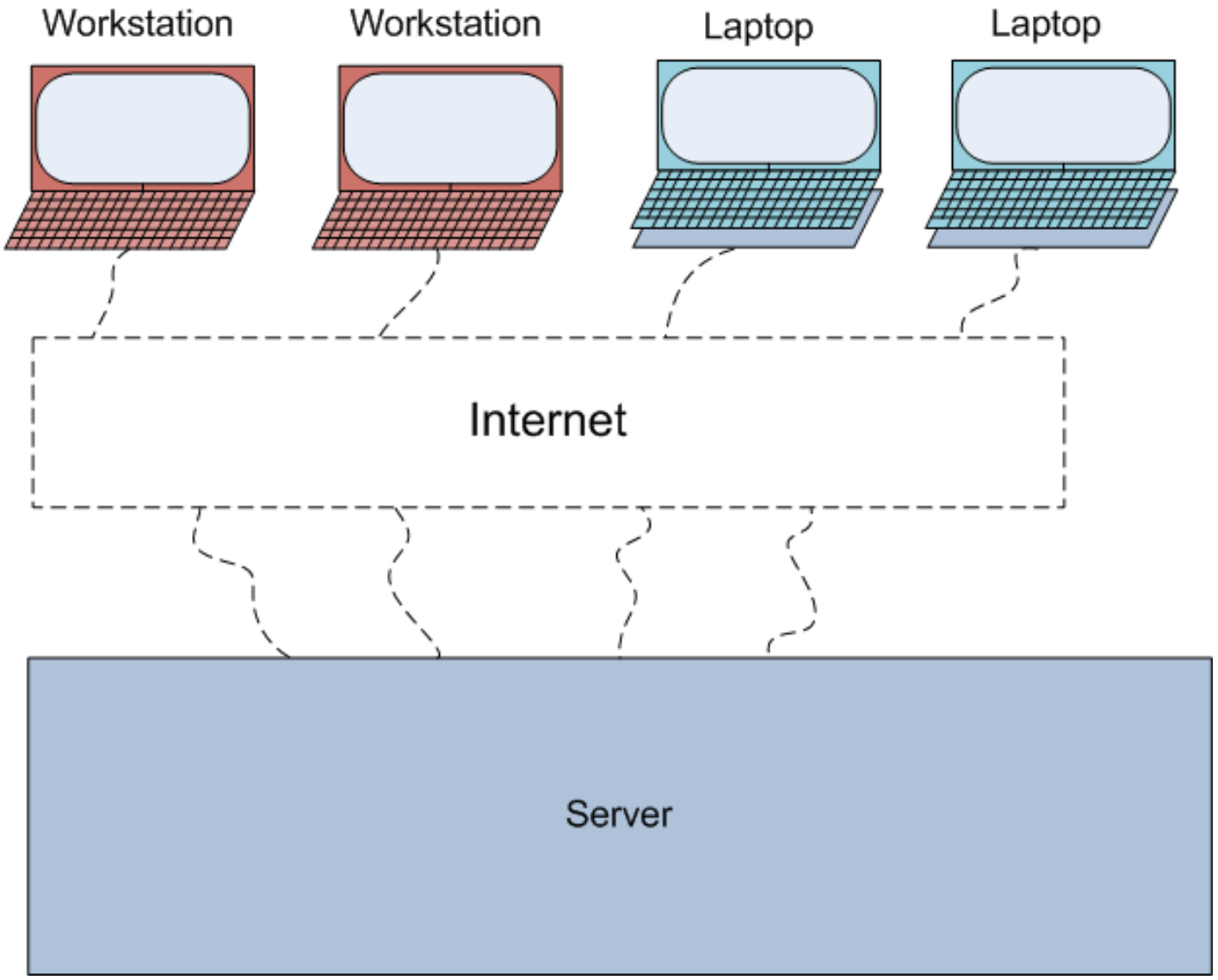
1990's-present

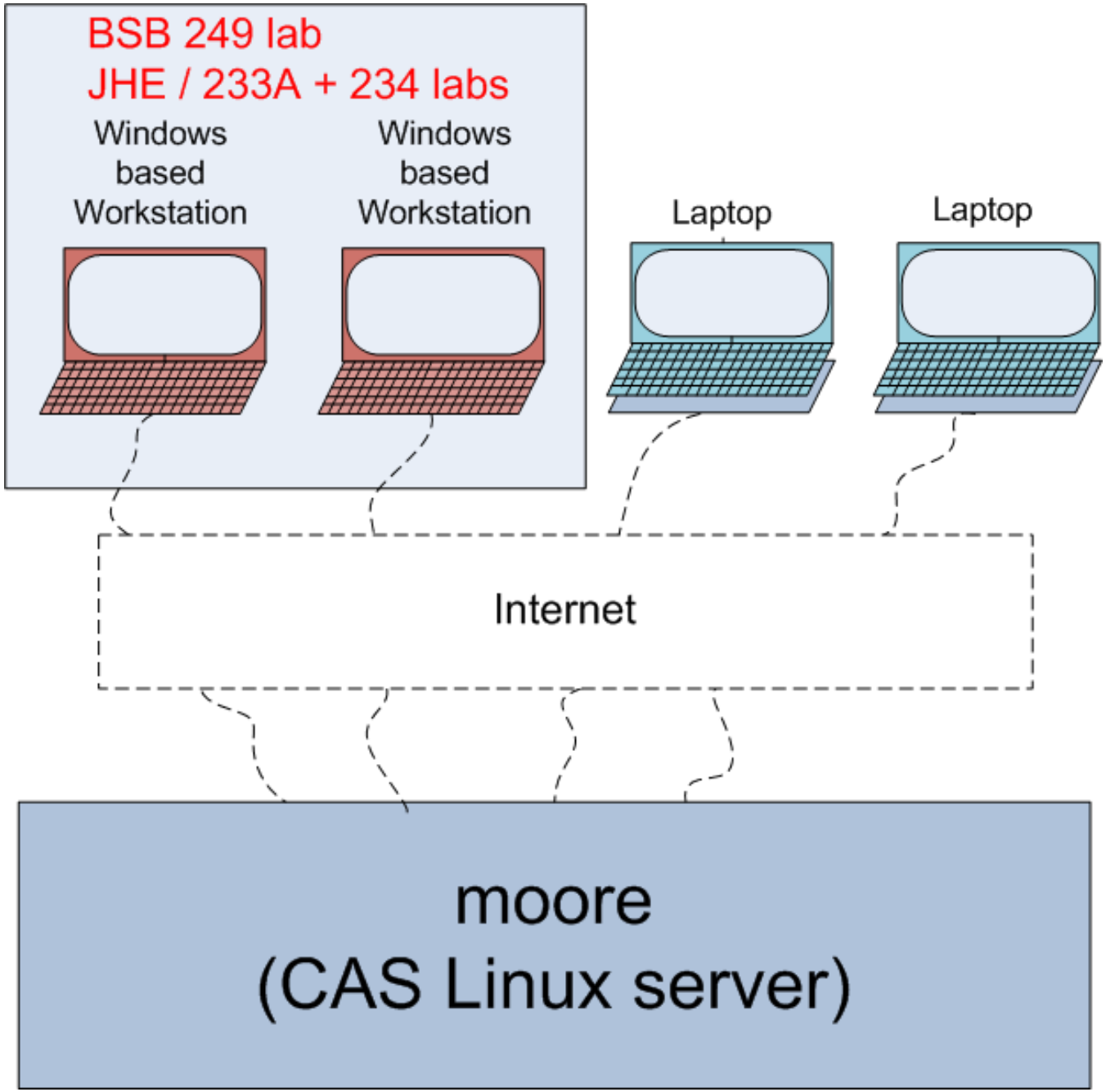


1970's

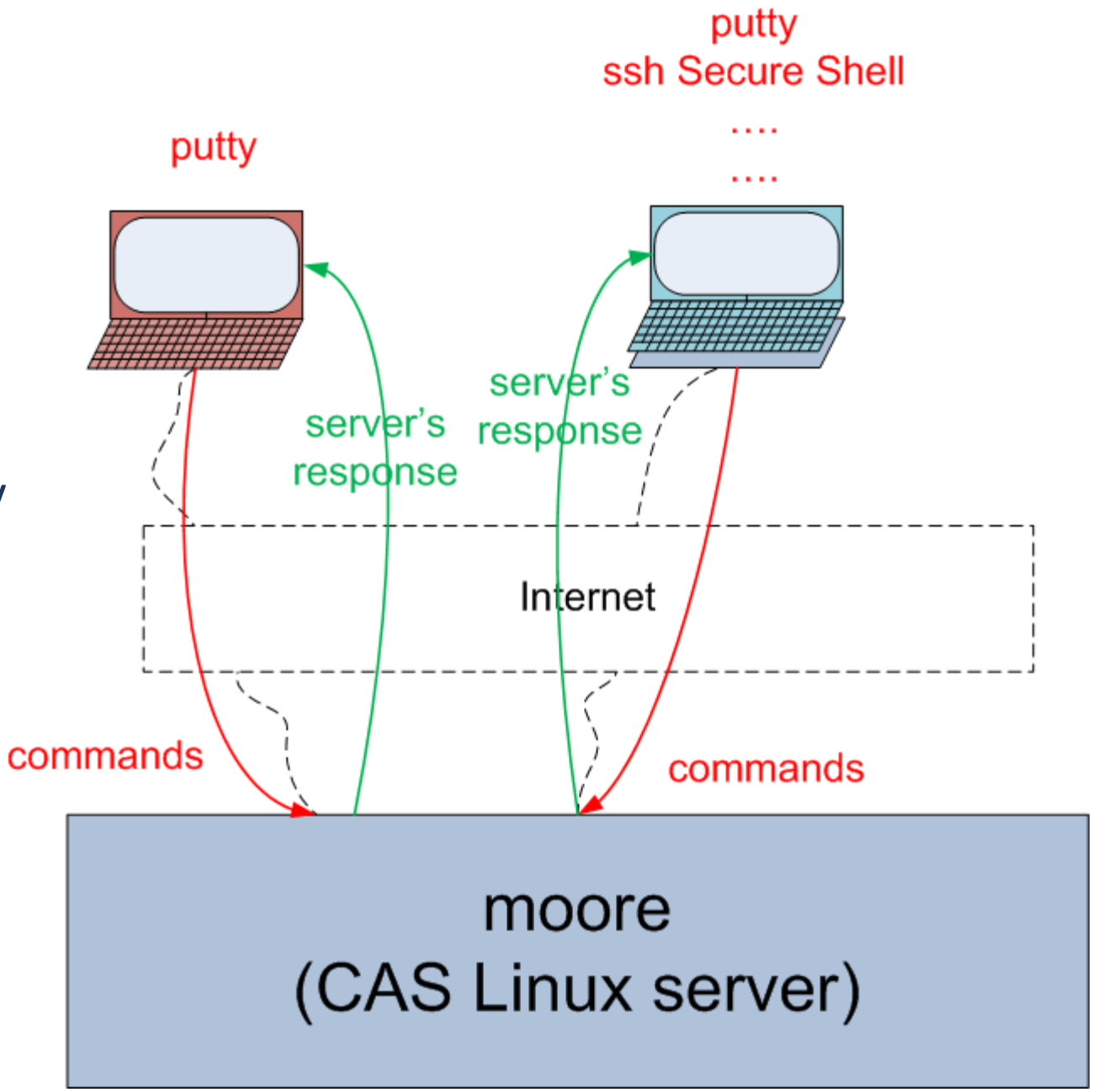


1990's - present

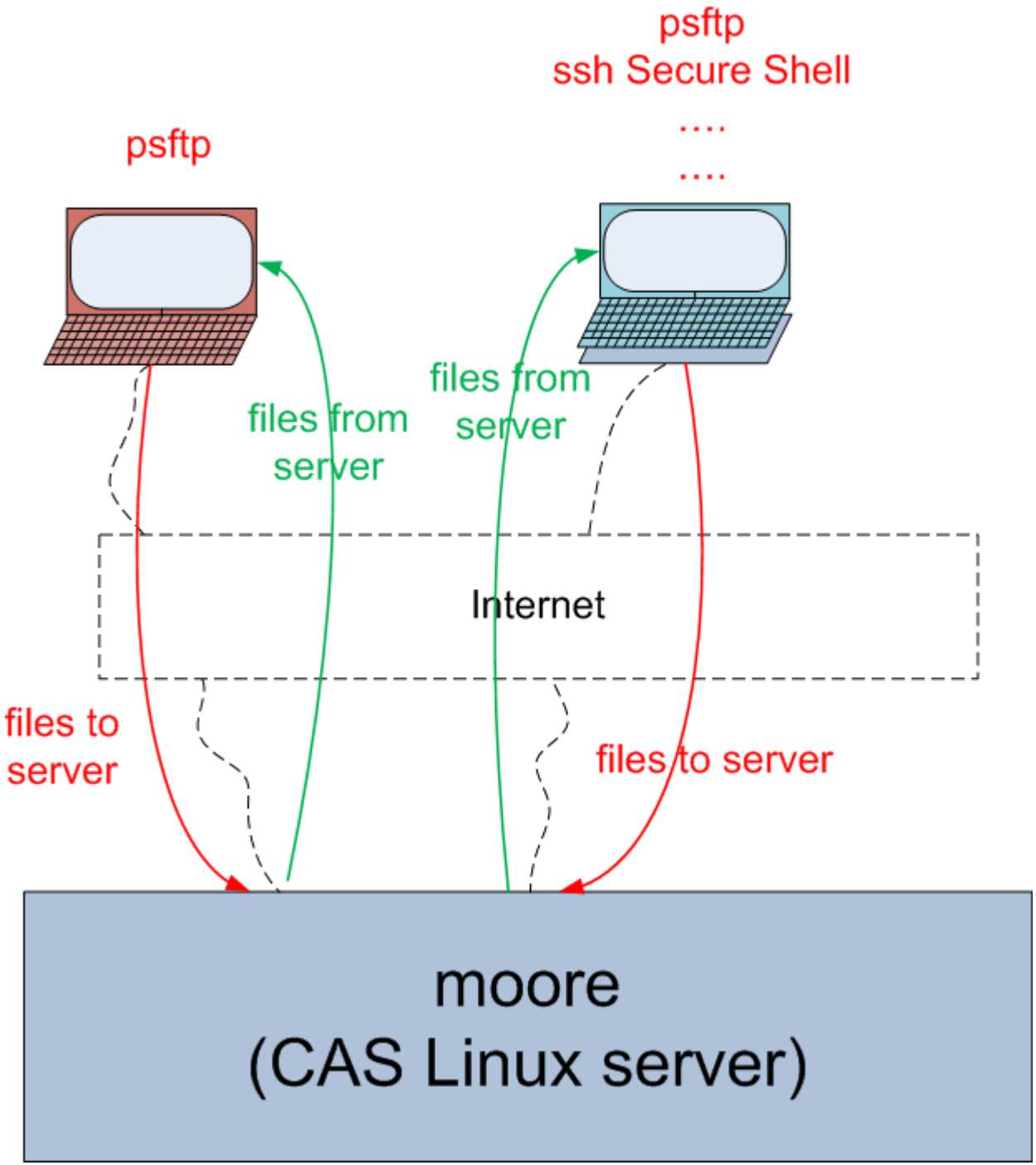



BSB 249 lab
JHE / 233A + 234 labs
Windows based Workstation
Windows based Workstation
Laptop
Laptop
Internet
moore
(CAS Linux server)

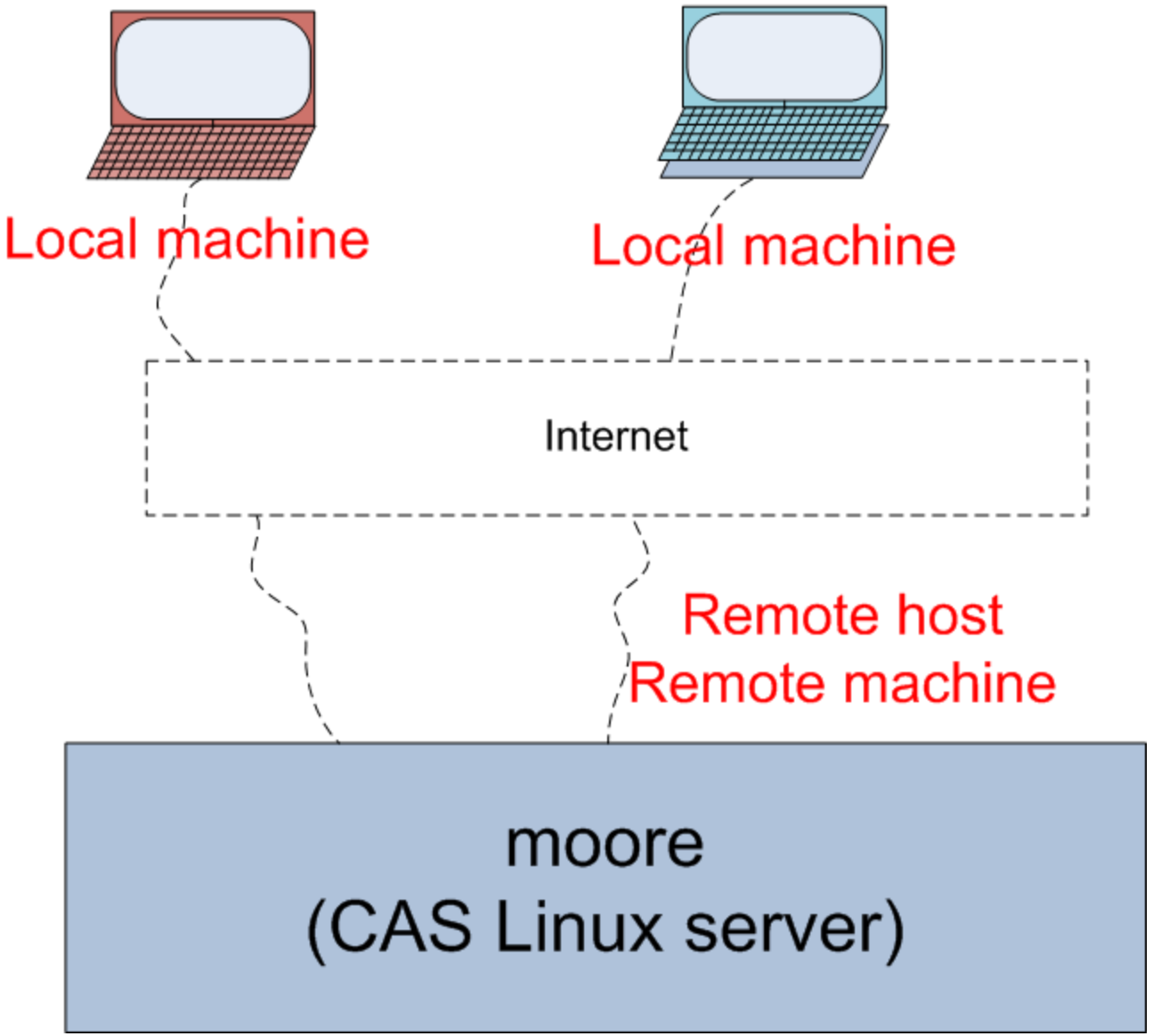
Requires:
a terminal emulation software (colloquially **terminal**) for sending commands to the server and brining back the server responses



Requires:
A file transfer software (colloquially **ftp**) for moving files to the server and from the server



Terminology



What is file?

- A logical unit to be manipulated as a whole (*like a book*) storing information (can be ***created***, ***destroyed***, ***named*** and ***renamed***, and ***moved***).

- A string of bits encoding the information (*a book is also a string of letters*).

Just like a book a file can be opened (***open***) and the information stored in it can be read (***read***), or modified, or new information inserted (***write***) and then the file can be closed (***close***) to make the changes permanent.

1 BIT (BInary digiT) is the least amount of information, and hence the least amount of storage (memory).

Think a simple switch:

ON	OFF
1	0

The same way 1000 meters is for brevity and convenience referred to as Kilometer, we refer to a group of 8 bits as 1 BYTE

Why 8 and not 7 or 9 bits per byte?

Why 12 inches to a foot ????
Why 3 feet in a yard ????
Why 1760 yards in a mile ????

Historical reasons, nothing else. The same for BYTE. 7 bits were enough to code for all telex characters (the utility reason, 128 codes were quite sufficient). 8 bits can give twice as many codes (256) and were deemed amply sufficient.

1 Kilobit = 2^{10} bits = 1024 bits
1 Kilobyte (KB) = 2^{10} bytes = 1024 bytes
(1 KB ~ 10^3 bytes)

1 Megabyte (MB) = 2^{10} KB = 2^{20} bytes = 1,048,576 bytes
(1 MB ~ 10^6 bytes)

1 Gigabyte (GB) = 2^{10} MB = 2^{20} KB = 2^{30} bytes = 1,073,741,824 bytes
(1 GB ~ 10^9 bytes)

1 Terabyte (TB) = 2^{10} GB = 2^{20} MB = 2^{30} KB = 2^{40} bytes = 1,099,511,627,776 bytes
(1TB ~ 10^{12} bytes)

1 Petabyte (PB) = 2^{10} TB = 2^{20} GB = 2^{30} MB = 2^{40} KB = 2^{50} bytes = 1,125,899,906,842,624 bytes
(1 PB ~ 10^{15} bytes)

Bits in different media:

- magnetic (hard disk): magnetized/demagnetized -- non-volatile
- optical (DVD): a hole/not a hole -- non-volatile
- flash (USB key): electronic NAND or NOR circuits – non-volatile
- main memory: electronic flip-flops -- volatile

How is information encoded by bit sequences ?

In many ways, but we are mainly interested in how ordinary characters (a .. z, A .. Z, 0 .. 9) and ordinary symbols ($, ? ...) are encoded.

ASCII table

ASCII TABLE

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]*

ASCII text characters:

9	Horizontal tab	\t
10	Line feed (end of line)	\n
13	Carriage return (end of line)	\r

DOS text files: end of line \r\n

UNIX text files: end of line \n

ASCII text characters:

9, 10, 13, 32 Space .. 126 -

So, roughly speaking the first 126 codes of 8 bit sequences, i.e. all sequences starting with 0 bit.

For binary codes see:

https://www.rapidtables.com/code/text/ascii-table.html

File is thus a sequence of bits encoding some information, or more commonly expressed as a sequence of bytes (or KB, MB, GB ...) the same way you would not describe the distance from Hamilton to Toronto in meters.

Format of a file is:

(a) text file (or more precisely ASCII text file)
    if all bytes of the file are ASCII text characters.

(b) binary file otherwise.

Note, that for a format of a file, its name does not matter!!!!!!!!!

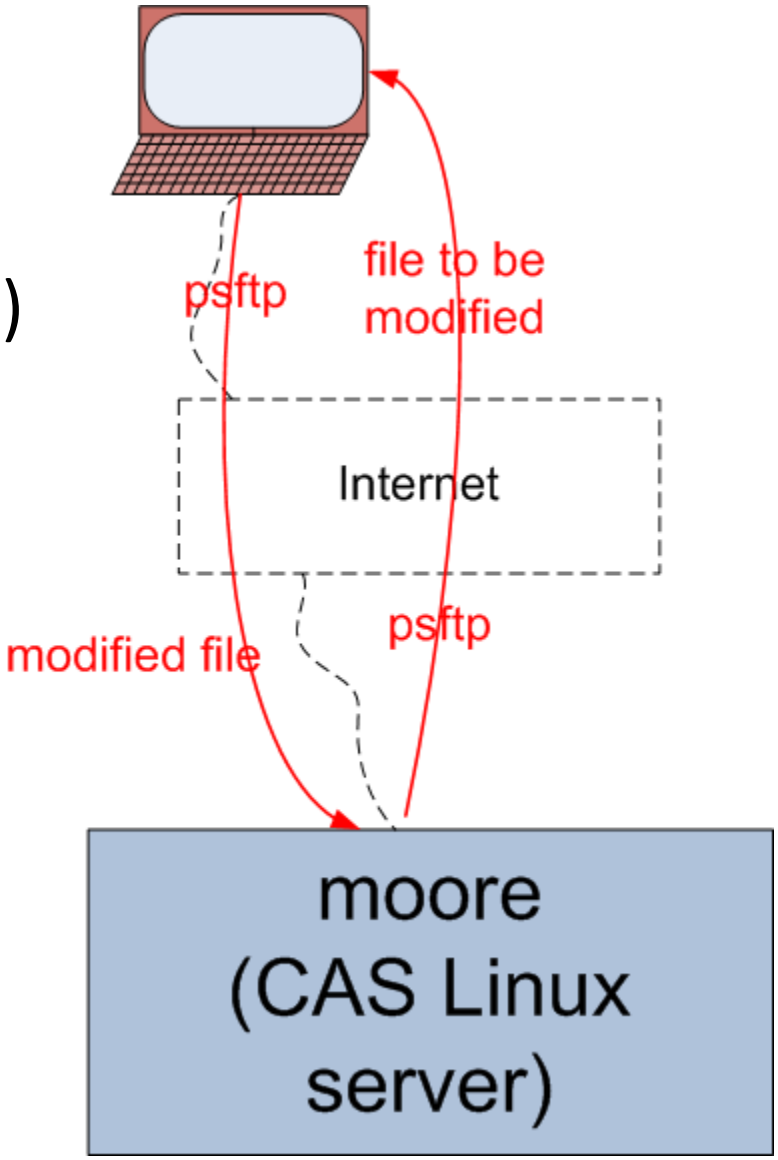
In DOS (and then in Windows) it was customary (but not obligatory) to call text files with extension .txt
In Windows it can be hidden (curse Bill Gates for extensions in the first place and their hiding in the second place)

In UNIX, there is no association of the name of a file and the format of that file.

How editing can be done:

(1) ftp the file from the server to the workstation
(2) use notepad (wordpad, word …) to modify the file
(3) ftp the modified file from the workstation to the server

Pro: little learning involved
Con: awkward and slow



A better option is to learn some ***nano*** or ***vi*** text editors (see Help section) and do the editing directly at the server. Basic use of ***nano*** is very simple with a very short learning curve, ***vi*** is slightly more complex, a bit longer learning curve, though more versatile and apt for programming.

Pro: simple and fast, no ftp involved
Con: requires some extra learning