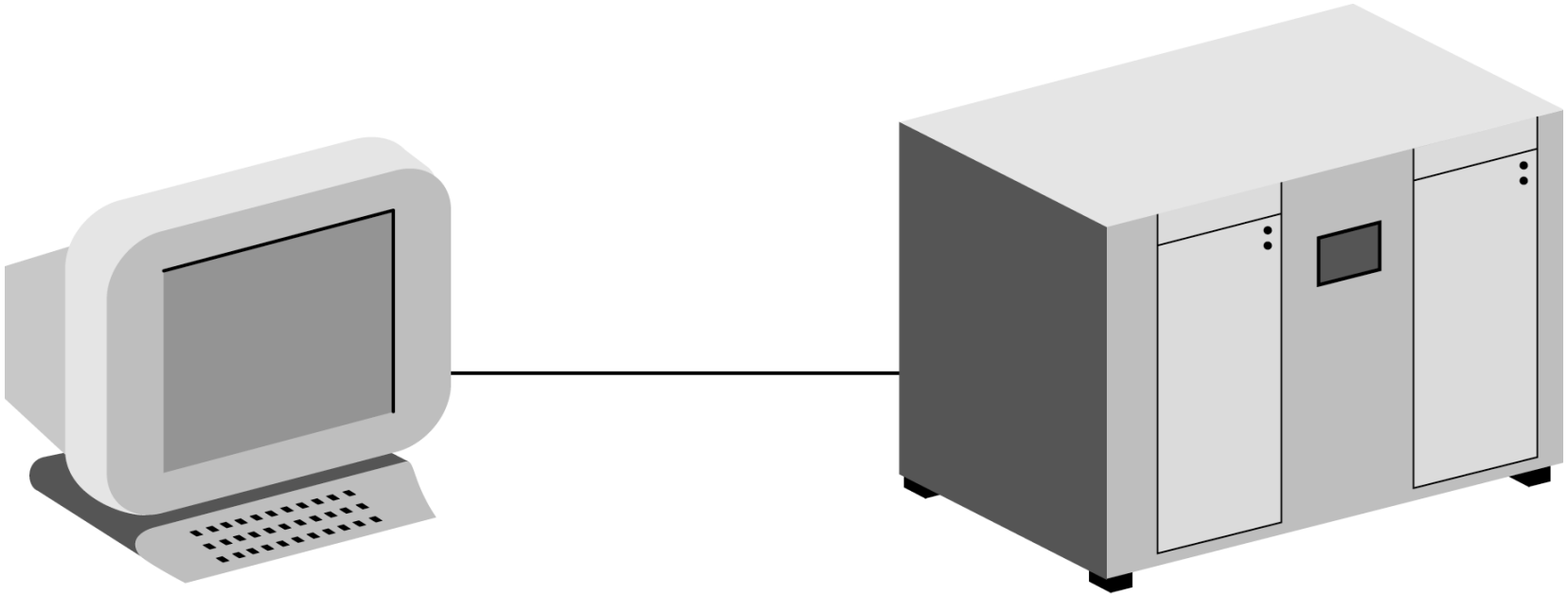
Basic Data Communications Concepts

OVERVIEW

- **Bits & Bytes**
- **Characters Codes**
- **Parallel vs. Serial**
- **Timing Methods for Serial Transmission**
- **Directionality of the Transmission Path**

FIGURE 3-1: TERMINAL AND HOST COMPUTER



A SIMPLE DATA COMM. MODEL

- **Host Computer - where the processing takes place**
  - **Mainframes**
  - **Minicomputers**
  - **Microcomputers (PCs)**
  - **Super Computers**
- **Terminal - a device that communicates with a host**
  - **Dumb - Send/Receive data. Can't modify data**
  - **Smart - Sends extra info to host (e.g., terminal ID)**
  - **Intelligent - Programmable**
- **PCs are the most "intelligent" terminal**
  - **Can emulate any of the three listed above.**
  - **Can also act as a host in some situations**

BITS & BYTES

- **With few exceptions, digital computers communicate through a series of 1's and 0's known as *bits*.**
- **This binary representation can also be thought of as being *on* and *off*.**
- **Groups of bits are referred to as *bytes***
  - **In most systems, a byte consists of 8 bits**
  - **Usually each byte represents a single character**
    - A-Z, a-z, 0-9
    - punctuation characters(e.g., @, #, %)
    - special characters (LF, CR, ESC)
- **Bits and bytes are closely related to the binary number system. See Appendix in text for more information**

CHARACTER CODES

- **The relationship of bytes to characters is determined by a *character code***
- **Each time a user presses a key on a terminal/PC, a binary code is generated for the corresponding character.**
- **Various character codes have been used in data communication including:**
  - **Morse, Baudot**
  - **EBCDIC, ASCII**
  - **Unicode**
- **Regardless of the character code, both the terminal/ host or sender/receiver must recognize the same coding scheme**

MORSE CODE

- First character code developed
- For transmitting data over telegraph wires
  - telegrams (remember Western Union)
- Used dots (short beep) and dashes (long beeps) instead of 1's and 0's
- More frequent the character, the fewer the beeps
- Problems:
  - variable "length" character representation
  - required pauses between letters
  - no lower case, few punctuation or special characters
  - no error detection mechanism

FIGURE 3-2: MORSE CODE

A	• –	N	– •	1	• – – – –
B	– • • •	O	– – –	2	• • – – –
C	– • – •	P	• – – •	3	• • • – –
D	– • •	Q	– – • –	4	• • • • –
E	•	R	• – •	5	• • • • •
F	• • – •	S	• • •	6	– • • • •
G	– – •	T	–	7	– – • • •
H	• • • •	U	• • –	8	– – – • •
I	• •	V	• • • –	9	– – – – •
J	• – – –	W	• – –	0	– – – – –
K	– • –	X	– • • –	.	• – • – • –
L	• – • •	Y	– • – –	,	– – • • – –
M	– –	Z	– – • •	?	• • – – • •

BAUDOT CODE

- **One of first codes developed for machine to machine communication**
- **Uses 1's and 0's instead of dots and dashes**
- **For transmitting telex messages (punch tape)**
- **Fixed character length (5-bits)**
  - **32 different codes**
  - **increased capacity by using two codes for shifting**
    - 11111 (32) Shift to Lower (letters)
    - 11011 (27) Shift to Upper (digits, punctuation)
  - **4 special codes for SP, CR, LF & blank**
  - **Total = 26 + 26 + 4 = 56 different characters**

BAUDOT CODE (cont.)

- **Problems:**
  - **required shift code to switch between character sets**
  - **no lower case, few special characters**
  - **no error detection mechanism**
  - **characters not ordered by binary value**
  - **designed for transmitting data, not for data processing**
- **International Baudot**
  - **Added a 6th bit for parity**
  - **Used to detect errors within a single character**

FIGURE 3-3: BAUDOT CODE

Character		Data bits				
Lower case	Upper case	5	4	3	2	1
A	_	0	0	0	1	1
B	?	1	1	0	0	1
C	:	0	1	1	1	0
D	$	0	1	0	0	1
E	3	0	0	0	0	1
F	!	0	1	1	0	1
G	&	1	1	0	1	0
H	#	1	0	1	0	0
I	8	0	0	1	1	0
J	'	0	1	0	1	1
K	(	0	1	1	1	1
L	)	1	0	0	1	0
M	.	1	1	1	0	0
N	,	0	1	1	0	0
O	9	1	1	0	0	0
P	0	1	0	1	1	0
Q	1	1	0	1	1	1
R	4	0	1	0	1	0
S	BELL	0	0	1	0	1
T	5	1	0	0	0	0
U	7	0	0	1	1	1
V	;	1	1	1	1	0
W	2	1	0	0	1	1
X	/	1	1	1	0	1
Y	6	1	0	1	0	1
Z	"	1	0	0	0	1
Letters (shift to Lower case column)		1	1	1	1	1
Figures (shift to Upper case column)		1	1	0	1	1
Space		0	0	1	0	0
Carriage return		0	1	0	0	0
Line feed		0	0	0	1	0
Blank		0	0	0	0	0

EBCDIC

- **Extended Binary Coded Decimal Interchange Code**
- **8-bit character code developed by IBM**
  - **used for data communication, processing and storage**
  - **extended earlier proprietary 6-bit BCD code**
  - **designed for backward compatibility or marketing?**
  - **still in use today on some mainframes and legacy systems.**
- **Allows for 256 different character representations (2^8)**
  - **includes upper and lower case**
  - **lots of special characters (non-printable)**
  - **lots of blank (non-used codes)**
    - assigned to international characters in various versions
  - **used with/without parity (block transmissions)**

FIGURE 3-4: EBCDIC CODE

Bit 0 1 2 3	4 5 6 7	0 0 0 0	0 0 0 1	0 0 1 0	0 0 1 1	0 1 0 0	0 1 0 1	0 1 1 0	0 1 1 1	1 0 0 0	1 0 0 1	1 0 1 0	1 0 1 1	1 1 0 0	1 1 0 1	1 1 1 0	1 1 1 1
0 0 0 0		NUL	SOH	STX	ETX	PF	HT	LC	DEL				VT	FF	CR	SO	SI
0 0 0 1		DLE	DC1	DC2	DC3	RES	NL	BS	IL	CAN	EM			IFS	IGS	IRS	IUS
0 0 1 0				FS		BYP	LF	EOB	PRE			SM			ENQ	ACK	BEL
0 0 1 1				SYN		PN	RS	UC	EOT					DC4	NAK		SUB
0 1 0 0		SP										¢	.	<	(	+	|
0 1 0 1		&										!	$	*	)	;	¬
0 1 1 0		-	/									¦	,	%	-	>	?
0 1 1 1											\	:	#	@	'	=	"
1 0 0 0			a	b	c	d	e	f	g	h	i						
1 0 0 1			j	k	l	m	n	o	p	q	r						
1 0 1 0			~	s	t	u	v	w	x	y	z						
1 0 1 1																	
1 1 0 0		{	A	B	C	D	E	F	G	H	I						
1 1 0 1		}	J	K	L	M	N	O	P	Q	R						
1 1 1 0				S	T	U	V	W	X	Y	Z						
1 1 1 1		0	1	2	3	4	5	6	7	8	9						□

Note: To read this chart, simply find the character on the chart, then look to the left side of the row for bits 0, 1, 2, and 3, and to the top of the column for bits 4, 5, 6, and 7. This is only one of many possible implementations of EBCDIC.

EBCDIC special characters

ACK	Acknowledgement	EOT	End of Transmission	PF	Punch Off
BEL	Bell	ETX	End of Text	PN	Punch On
BS	Backspace	FF	Form Feed	PRE	Prefix
BYP	Bypass	FS	File Separator	RES	Restore
CAN	Cancel	HT	Horizontal Tab	RS	Reader Stop
CR	Carriage Return	IFS	Information File Separator	SI	Shift In
DC1	Device Control 1	IGS	Information Group Separator	SM	Start Message
DC2	Device Control 2	IL	Idle	SO	Shift Out
DC3	Device Control 3	IRS	Information Record Separator	SOH	Start of Heading
DC4	Device Control 4	IUS	Information Unit Separator	SP	Space
DEL	Delete	LC	Lower Case	STX	Start of Text
DLE	Data Link Escape	LF	Line Feed	SUB	Substitute
EM	End of Medium	NAK	Negative Acknowledgement	SYN	Synchronous Idle
ENQ	Enquiry	NL	New Line	UC	Upper Case
EOB	End of Block	NUL	Null	VT	Vertical Tab

ASCII CODE

- American Standard Code for Information Interchange
- 7-bit code developed by the American National Standards Institute (ANSI)
  - most popular data communication character code today
- Allows for 128 different character representations (2^7)
  - includes upper and lower case
  - lots of special characters (non-printable)
  - generally used with an added parity bit
  - better binary ordering of characters than EBCDIC
- Extended ASCII uses 8 data bits and no parity
  - Used for processing and storage of data
  - Allows for international characters
  - 8th bit stripped of for transmission of standard character set

FIGURE 3-5: 7-BIT ASCII CODE

Bits 7654321	Character	Bits 7654321	Character	Bits 7654321	Character	Bits 7654321	Character
0000000	NUL	0100000	SP	1000000	@	1100000	`
0000001	SOH	0100001	!	1000001	A	1100001	a
0000010	STX	0100010	“	1000010	B	1100010	b
0000011	ETX	0100011	#	1000011	C	1100011	c
0000100	EOT	0100100	$	1000100	D	1100100	d
0000101	ENQ	0100101	%	1000101	E	1100101	e
0000110	ACK	0100110	&	1000110	F	1100110	f
0000111	BEL	0100111	'	1000111	G	1100111	g
0001000	BS	0101000	(	1001000	H	1101000	h
0001001	HT	0101001	)	1001001	I	1101001	i
0001l0	LF	0101010	*	1001010	J	1101010	j
0001011	VT	0101011	+	1001011	K	1101011	k
0001100	FF	0101100	,	1001100	L	1101100	1
0001101	CR	0101101	-	1001101	M	1101101	m
0001110	SO	0101110	.	1001110	N	1101110	n
0001111	SI	0101111	/	1001111	O	1101111	o
0010000	DLE	0110000	0	1010000	P	1110000	p
0010001	DC1	0110001	1	1010001	Q	1110001	q
0010010	DC2	0110010	2	1010010	R	1110010	r
0010011	DC3	0110011	3	1010011	S	1110011	s
0010100	DC4	0110100	4	1010100	T	1110100	t
0010101	NAK	0110101	5	1010101	U	1110101	u
0010110	SYN	0110110	6	1010110	V	1110110	v
0010111	ETB	0110111	7	1010111	W	1110111	w
0011000	CAN	0111000	8	1011000	X	1111000	x
0011001	EM	0111001	9	1011001	Y	1111001	y
0011010	SUB	0111010	:	1011010	Z	1111010	z
0011011	ESC	0111011	;	1011011	[	1111011	{
0011100	FS	0111100	<	1011100	\	1111100	|
0011101	GS	0111101	=	1011101	]	1111101	}
0011110	RS	0111110	>	1011110	^	1111110	~
0011111	US	0111111	?	1011111	—	1111111	DEL

FIGURE 3-5: ASCII NON-PRINTABLE CODES

ASCII control characters			
BEL	Bell	EM	End of Medium
CAN	Cancel	ESC	Escape
DC1	Device Control 1	NUL	Null
DC2	Device Control 2	SI	Shift In
DC3	Device Control 3	SO	Shift Out
DC4	Device Control 4	SUB	Substitute
DEL	Delete		
Control codes			
ACK	Acknowledge	ETX	End of Text
DLE	Data Link Escape	NAK	Negative Acknowledge
ENQ	Enquiry	SOH	Start of Heading
EOT	End of Transmission	STX	Start of Text
ETB	End of Transmission Block	SYN	Synchronous Idle
Format effectors			
BS	Backspace	HT	Horizontal Tabulation
CR	Carriage Return	LF	Line Feed
FF	Form Feed	VT	Vertical Tabulation
Information separators			
FS	File Separator	RS	Record Separator
GS	Group Separator	US	Unit Separator

UNICODE

- **Designed to support international languages:**

  Latin; Greek; Cyrillic; Armenian; Hebrew; Arabic; Syriac; Thaana; Devanagari; Bengali; Gurmukhi; Oriya; Tamil; Telegu; Kannada; Malayalam; Sinhala; Thai; Lao; Tibetan; Myanmar; Georgian; Hangul; Ethiopic; Cherokee; Canadian-Aboriginal Syllabics; Ogham; Runic; Khmer; Mongolian; Han (Japanese, Chinese, Korean ideographs); Hiragana; Katakana; Bopomofo and Yi

- **Uses a 16-bit code for total of 65,536 possible char.**
  - **Incorporates ASCII in first 128 codes**
  - **Incorporates LATIN in first 256 codes**
- **Support found in newer hardware & software, especially web technologies (e.g., JAVA, XML, HTML)**
- **For more see www.unicode.org**

SUMMARY OF CHARACTER CODES

Morse	**=**	**.-**
Baudot	**=**	**5 bit (no parity)**
Int. Baudot	**=**	**6 bit (5 data + 1 parity)**
ASCII	**=**	**8 bit (7 data + 1 parity)**
EBCDIC	**=**	**9 bit (8 data + 1 parity)**
UNICODE	**=**	**16 bits (no parity)**

- **Normally terminals and hosts must use the same code**
- **However, code conversion hardware/software can be used to allow different machines to communicate**

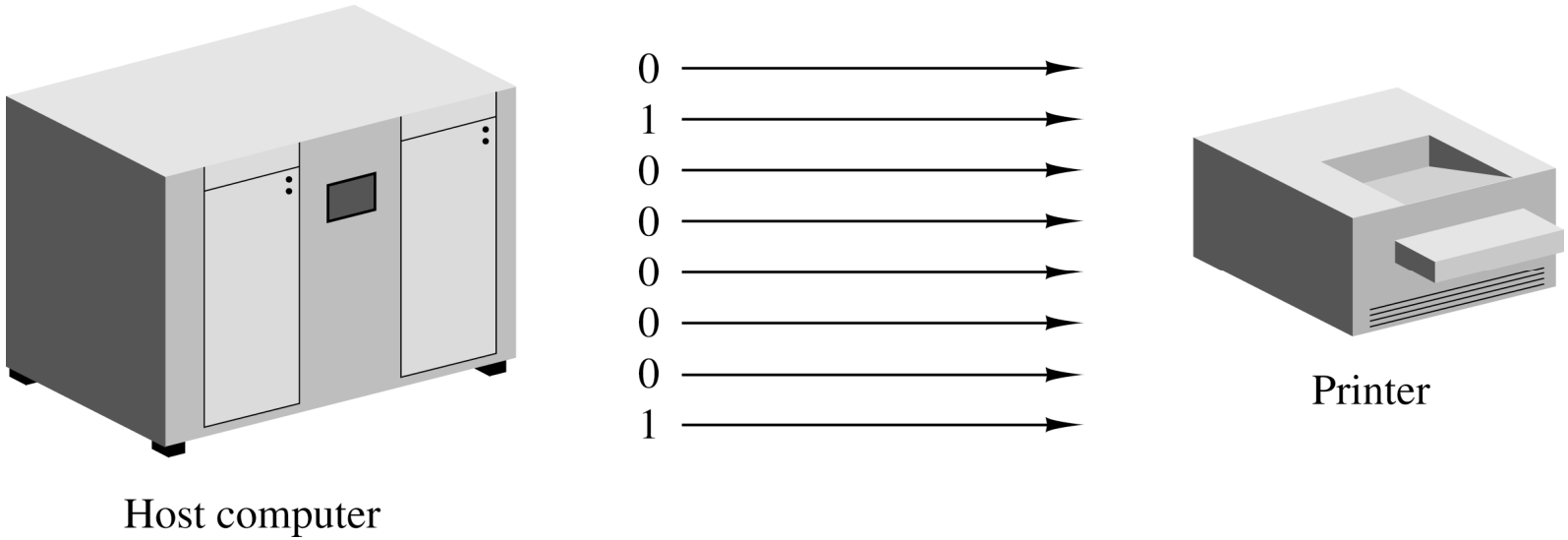
SUMMARY OF CHARACTER CODES (cont.)

- **Bits per character affect**
  - **storage requirements**
  - **throughput of information**
- **Use of larger codes became feasible due to**
  - **higher transmission speeds**
  - **denser storage mediums**
- **Choice of character coding scheme is a trade off between**
  - **simplicity & brevity**
  - **expressivity**

TRANSMISSION CHARACTERISTICS

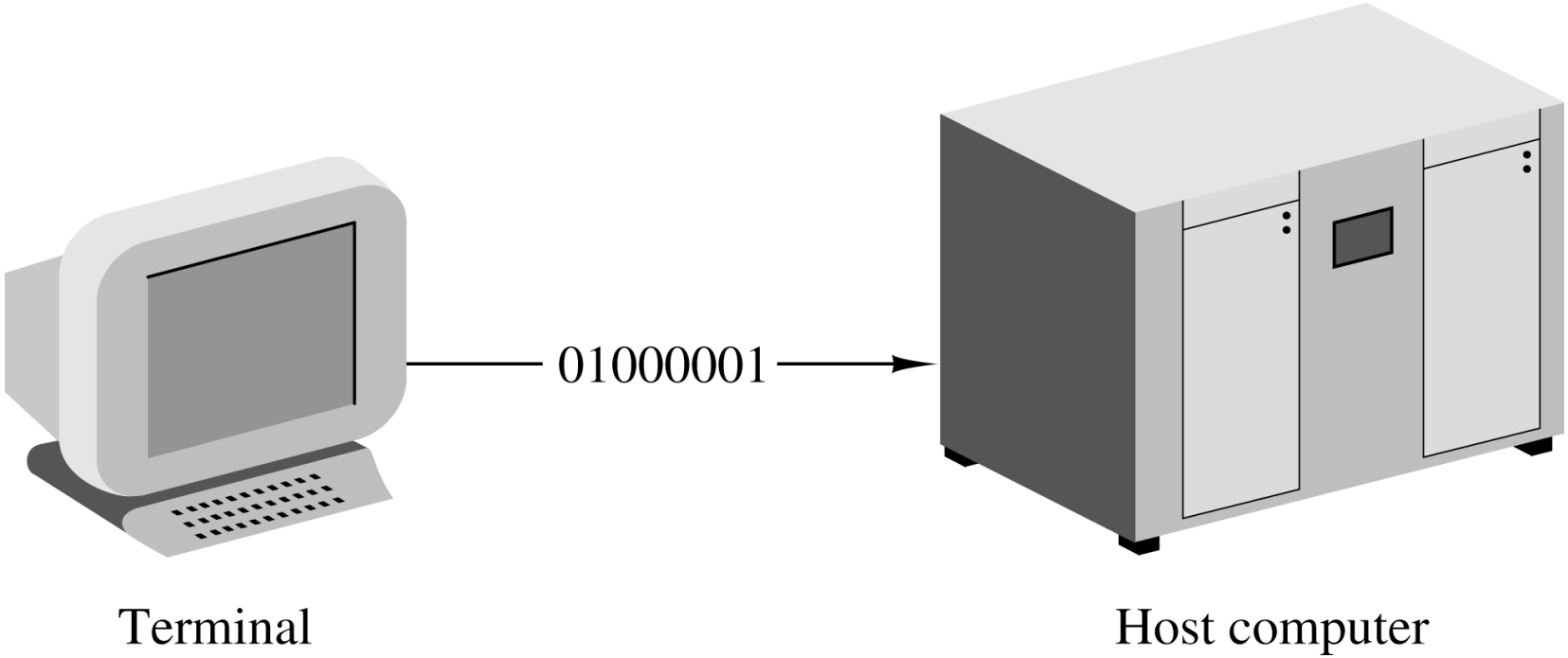
- **A character code determines what bits we will send between a terminal and host**
- **But how will those bits be sent:**
  - **Parallel vs. Serial Transmission**
  - **Serial Transmission Timing**
  - **Direction of Transmission Path**
  - **Others which we'll look at later**
    - speed
    - organization of data (protocol)
    - transmission media

FIGURE 3-8: PARALLEL DATA TRANSMISSION



Used most often for communication with local devices

FIGURE 3-9: SERIAL DATA TRANSMISSION

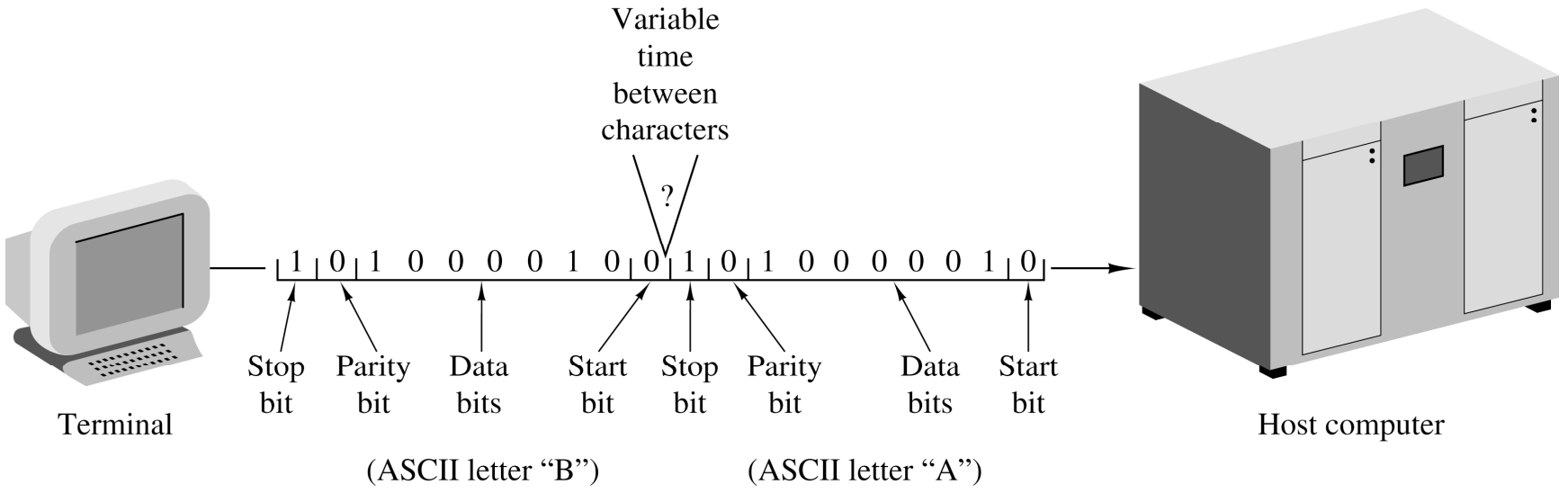


Used most often for data communication

SERIAL TRANSMISSION TIMING

- **ASYNCHRONOUS**
  - **Bits comprising a character are transmitted independent of timing of any other character**
  - **Asynchronous information bits are preceded by a start bit and followed by a stop bit**
    - Start bit is always a 0 (space)
    - Stop bit is always a 1 (mark)
  - **Start and Stop bits along with a parity bit result in 30% overhead in data transmission using ASCII codes**
  - **Also known as start-stop transmission**
  - **Used with dumb terminals and “character at a time” applications**

FIGURE 3-10: ASYNCHRONOUS DATA TRANSMISSION



SERIAL TRANSMISSION TIMING (cont.)

- **SYNCHRONOUS**
  - **Timing between sending and receiving locations is synchronized for transmission using clocks or sync characters**
  - **Provides support for block-mode transmission of data**
  - **No start or stop bits**
  - **Parity bits may not be used depending on the character code and block protocol**
  - **Frequently used with smart/intelligent terminals**
    - User inputs data which is held in the terminal
    - After input is complete, user presses the “send” or “enter” key and all data is transmitted in a single block.

FIGURE 3-11: SYNCHRONOUS DATA TRANSMISSION

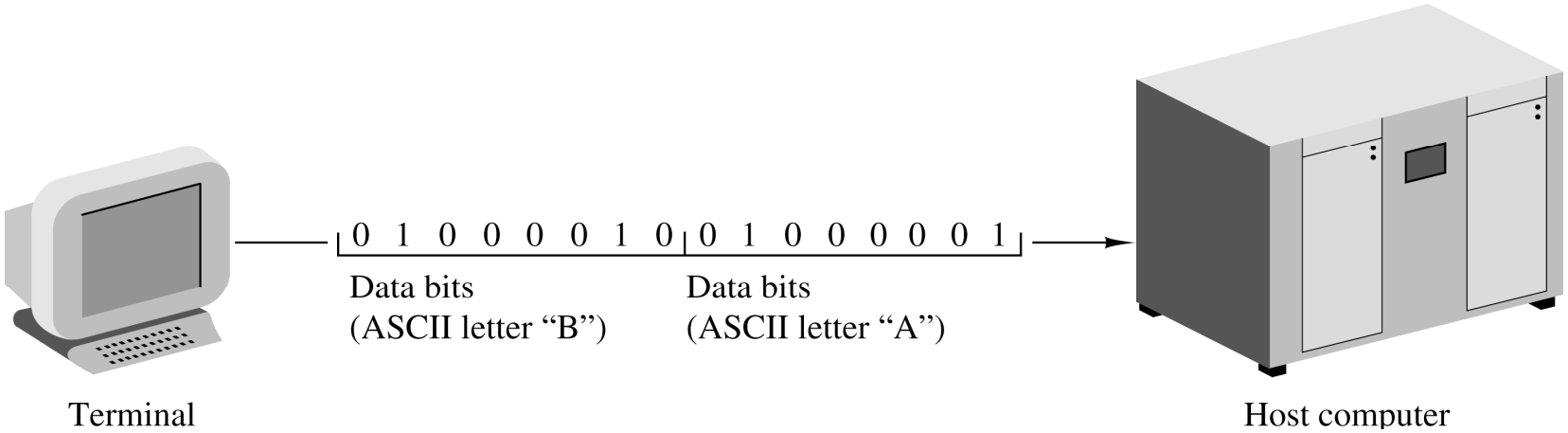


FIGURE 3-12: SYNCHRONOUS AND ASYNCHRONOUS TRANSMISSION EFFICIENCY

Asynchronous transmission efficiency	
ABCDEFGHIJKLMNOPQRSTUVWXYZ, etc., ABCDEFGHIJKLMNOPQRSTUVWXYZ	
Each character = 8 data bits, 1 start bit, 1 stop bit = 10 total bits	
If we need to send 1000 characters, then 10000 total bits are sent, of which 8000 are data bits	→80% efficiency
If we need to send 40 characters, then 400 total bits are sent, of which 320 are data bits	→ 80%efficiency
If we need to send 20 characters, then 200 total bits are sent, of which 160 are data bits	→80% efficiency

Synchronous transmission efficiency	
*****ABCDEFGHIJKLMNOPQRSTUVWXYZ, etc., ABCDEFGHIJKLMNOPQRSTUVWXYZ*****	
Assume that each block of data needs 10 special characters (shown as *) Each character = 8 data bits	
If we need to send 1000 characters, we add the 10 special characters, for a total of 1010 characters, so we send 8080 total bits, of which 8000 are data bits	→99. 1% efficiency
If we need to send 40 characters, we add the 10 special characters, for a total of 50 characters, so we send 400 total bits, of which 320 are data bits	→80% efficiency
If we need to send 20 characters, we add the 10 special characters, for a total of 30 characters, so we send 240 total bits, of which 160 are data bits	→66.7% efficiency

FIGURE 3-13: SIMPLEX COMMUNICATIONS PATH

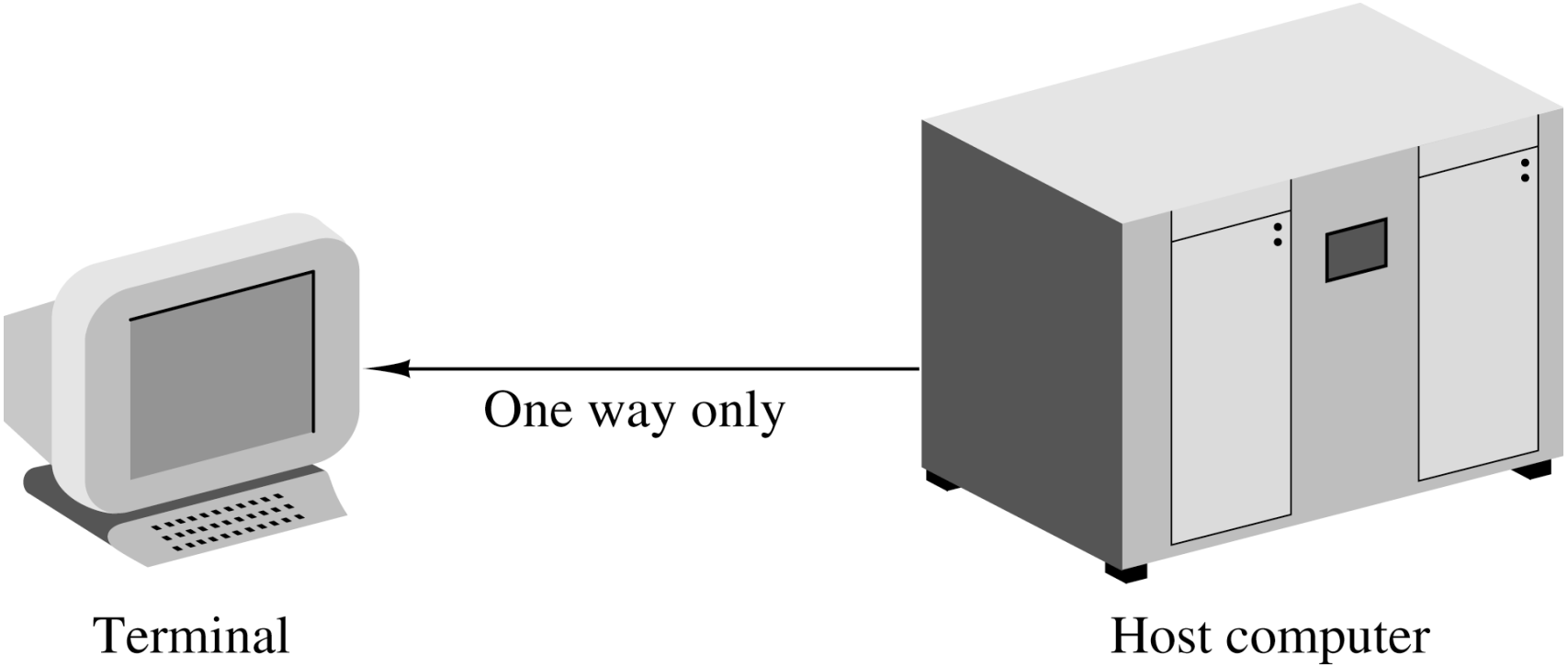


FIGURE 3-14: HALF DUPLEX COMMUNICATIONS PATH

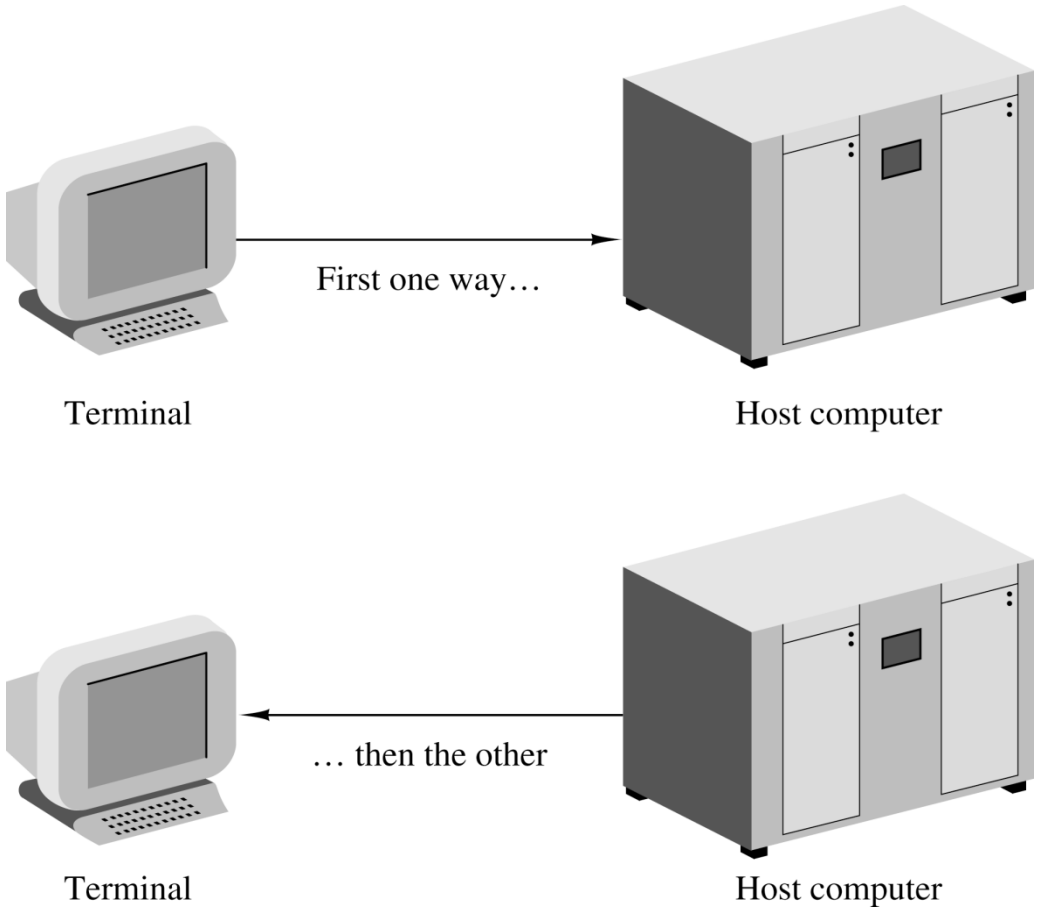


FIGURE 3-15: FULL DUPLEX COMMUNICATIONS PATH