PACTOR:
An Overview of a New and Effective HF Data Communication Protocol

Gwyn Reedy, W1BEL

Data communication via amateur radio using HF frequencies has recently become more effective and enjoyable due to a new communication protocol called PACTOR. PACTOR was developed by two enterprising German amateurs, DL6MAA and DF4KV. This article is based on the information provided by these gentlemen in their various writings.

PACTOR Features

PACTOR was designed to overcome the shortcomings exhibited by both packet and AMTOR in HF operation while remaining affordable for the average amateur operator.

- Error-free data transmission (less than 1 x 10-S)
- True binary data transmission
- Efficient use of channel capacity
- Good interference tolerance
- Requires only 600 Hz channel bandwidth
- Complete visibility of sending and receiving callsigns

Why PACTOR?

HF propagation is characterized by multipath propagation which induces ‘bit stretching’ and phase distortions, fading, impulse noise, and interference by other stations, among other obstacles to communication.

The PACTOR mode is similar to AMTOR which is good for ordinary HF communication. Both use half duplex ARQ; packets (blocks) of data carrying the information are acknowledged with short ‘receipt’ signals by the receiving station. When errors occur, the receiver can request the repetition of a packet with relevant control signals.

PACTOR uses a MASTER/SLAVE phasing like AMTOR. The SLAVE clock is synchronized to the MASTER timing. Only the MASTER corrects his Receive phase.

A long series of tests conducted by DL6MAA and DF4KV have shown that for operation in rapidly changing conditions, it is not a good policy to adjust packet length automatically. Simulations and on-the-air testing showed the optimum HF packet length to be about one second. To compensate for varying conditions, PACTOR varies the number of data characters in the data block, but does not change any of the synchronous timing parameters. PACTOR determines the proper baud rate to use based on the accuracy of bit transitions and the link error statistics.

Data blocks have CRC-16 checking as is done in AX25 packet. This is much more robust than the parity bits FEC used in AMTOR.

The data field of the PACTOR packet can contain any digital information; the format of the codes is specified in the status byte. At the present time the choice is between 8 bit ASCII and Huffman compressed 7 bit ASCII.

Authorization

The US FCC regulations specify that Baudot or ASCII codes may be used for data transmission, The 8 bit ASCII text transmitted by PACTOR is closer to ‘pure ASCII’ than the bit-stuffed HDLC used by AX25 packet, so there should be no question of the legality of this mode. The compressed PACTOR data mode uses ASCII characters encoded in a channel-capacity-enhancing format which still meets the intent of the regulations. The regulations regarding amateur transmissions require that there be no intent to obscure the data transmitted. The Huffman encoding scheme is published as an appendix to this article. It seems reasonable to me to consider that encoding for spectrum efficiency using a widely published encoding scheme shows no intent to encrypt the content of the data transmission and is therefore allowable under US regulations.

Performance

PACTOR achieves good throughput during poor HF conditions by a variety of techniques. The actual baud rate is kept low - the same as AMTOR. This is one third the rate of typical HF AX.25 packet operation. From another point of view, AMTOR and PACTOR bits are three times as long as 300 baud HF packet bits, thus providing much increased protection from the bit smearing caused by multipath propagation,

A doubling of the throughput compared to AMTOR results from sending longer blocks of data (but still short enough to cope with most fades) thus reducing the percentage of overhead carried. In addition, the ability to automatically double the data content of each block under favorable conditions provides a considerable increase in efficiency.

Finally, encoding ASCII text (7 bit characters) using Huffman codes increases throughput by an average of at least 70 percent.

Memory-ARQ

A significant feature of PACTOR is Memory-ARQ. Copies of the repeated reception of the same packet which fails the CRC are aggregated in memory and are summed individually for each bit. The aggregate of all unsuccessful transmissions is decoded which effectively increases the signal to noise ratio by about 15 dB. This PACTOR feature is hardware dependent and prevents the proper implementation of PACTOR as a software- only upgrade to packet or AMTOR equipment.

The combination of the above factors provides a protocol which can provide a throughput nearly equal to HF packet in the best of conditions, and much better throughput than packet during typical conditions. Compared to AMTOR, the throughput in good conditions is up to four times as great. During the poorest of conditions, throughput is considerably better than AMTOR because of the **CRC-16** error checking and Memory-ARQ capabilities.

Appendix: PACTOR Huffman code

Huffman coding is relatively indifferent to differences between red and theoretical alphabet character frequencies, so that similar good results are obtained in German and English plain text. The compression factor attained with ASCII amounts to about 1.7, resulting in an average of 4.5 bits per character. A greater compression factor would require considering the statistical relationships between the individual characters (Markov encoding).

Code in order of frequency, LSB (sent first) on the left:

Character	ASCII	Huffman
space	32	10
e	101	011
n	110	0101
i	105	1101
r	114	1110
t	116	00000
s	115	00100
d	100	00111
a	97	01000
u	117	1111.1
l	108	000010
h	104	000100
g	103	000111
m	109	001011
<CR>	13	001100
<LF>	10	00'1101
o	111	010010
c	99	010011
b	98	0000110
f	102	0000111
w	119	0001100
D	68	0001101
k	107	0010101
z	122	1100010
.	46	1100100
,	44	1100101

S	83	11’11011
A	65	00’101001
E	69	11000000
p	112	11000010
v	118	11000011
0	48	11000111
F	70	11001100
B	66	11001111
C	67	lllloool
I	73	11’110010
T	84	11’110100
0	79	000101000
P	80	000101100
1	49	001010000
R	82	110000010
(	40	110011011
)	41	110011100
L	76	110011101
N	78	111100000
Z	90	111100110
M	77	111100110
9	57	0001010010
W	87	0001010100
5	53	~01010101
Y	121	0001010110
2	50	0001011010
3	51	0001011011
4	52	0001011100
6	54	0001011101
7	55	0001011110
8	56	0001011111
H	72	0010100010
J	74	1100000110
U	85	1100000111
V	86	1100011000
<FS>	28	1100011001
x	120	1100011010
K	75	1100110100
3	63	1100110101
=	61	1111000010
4	113	1111010110
Q	81	1111010111
J	106	00010100110
G	71	00010100111
-	45	00010101111
	58	00101000111
!	33	11110011101
/	47	11110011110
*	42	001010001100
	34	110001101100
	37	110001101101
	39	110001101110
	95	111100001100
&	38	111100111001
+	43	111100111110
>	62	111100111111
@		0001010111000
	36	0001010111001
<	60	0001010111010
X	88	0001010111011
#	35	0010100011011
Y	89	00101000110101
	59	111100001’10100
	93	11110000110101
[	91	001010001101000
]	93	001010001101001
	127	110001101111000
~	126	110001101111001
}	125	110001101’1111010
	124	110001101’111011
{	123	110001101111100
’	96	110001101111101
^	94	110001101111110
<US>	32	110001101111111
<GS>	29	111100001101100
<ESC>	27	111100001101101
<EM>	25	111100001101110
<CAN>	24	111100001101111
<ETB>	23	111100001110000
<SYN>	22	111100001110001
<NAK>	21	111100001110010
<DC4>	20	111100001110011
<DC3>	19	111100001110100
<DC2>	18	111100001110101
<DC1>	17	111100001110110
<DLE>	16	111100001110111
<RS>	30	111100001111000
<SI>	15	111100001111001
<SO>	14	111100001111010
<FF>	12	111100001111011
<VT>	11	111100001111100
<HT >	9	111100001111101
<BS>	8	111100001111110
<BEL>		111100001111111
<ACK>	6	1111.00112000000
<ENQ>	5	111100111000001
<EOT>	4	111100111000010
<ETX>	3	111100111000011
<STX>	2	111100111000100
<SOH>	1	111100111000101
<NUL>	0	111100111000110
<SUB>	26	111100111000111