

Industry Canada
Library - Queen
JUL 21 1998
Industrie Canada
Bibliothèque - Queen

MEMORY ARQ SCHEMES FOR HF DATA TRANSMISSION

FINAL REPORT

BY

ALBERTO LEON-GARCIA
DEPARTMENT OF ELECTRICAL ENGINEERING
UNIVERSITY OF TORONTO

31, MARCH 1985

PREPARED FOR
DEPARTMENT OF COMMUNICATIONS
UNDER CONTRACT OST84-00265

COMMUNICATIONS CANADA
MAY 22 1985
LIBRARY - BIBLIOTHÈQUE

P
91
C655
L396
1985

②
MEMORY ARG SCHEMES FOR HF DATA TRANSMISSION

FINAL REPORT

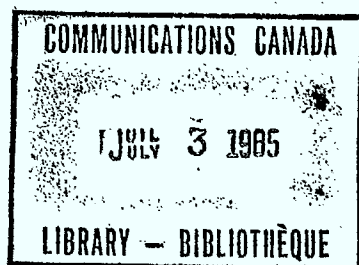
by

①
Alberto Leon-Garcia
Department of Electrical Engineering
University of Toronto

31 March, 1985

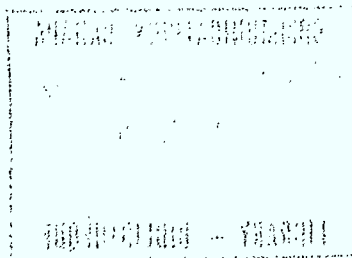
prepared for

Department of Communications
under Contract DST84-00265



P
91
C655
L396
1985

DD 5468198
DL 5468223



1. SUMMARY

The objective of this project was to investigate and develop memory ARQ schemes for improving the performance of multi-FSK data transmission over the HF radio channel. A set of experiments was conducted to evaluate the performance of various ARQ schemes. In this document we report on the design and setup of the experiments, and we present the experimental results for memory ARQ schemes without coding as well as with block and convolutional codes.

The report is organized as follows. In section 2 the transmission frame format is described, and a number of basic ARQ schemes are considered. Section 3 describes the experiments that were conducted to obtain the relative performance of the various ARQ protocols under equal transmission conditions. Section 4 presents performance results for memory ARQ protocols. In Section 5 a number of modifications of the basic ARQ schemes are considered. The corresponding performance results are then given in Section 6.

2. BASIC ARQ SCHEMES

The system under consideration is an outgrowth of the two-tone frequency diversity system used in the Mini-message terminal developed at the Department of Communications. The system utilizes a half-duplex error-control protocol in which frames of message subpackets are transmitted. Each frame consists of a header followed by 8 subpackets. Each subpacket has its own sequence number and CRC byte, so that subpackets are acknowledged and retransmitted separately. Efficiencies are thus attained by sharing the header overhead over many subpackets as well as by selectively retransmitting only the subpackets found in error. The objective of this project was to develop a higher throughput system by increasing the number of tones to 8 or 16 as well as by introducing a more sophisticated error-control protocol.

Figure 1a shows the structure of a single transmission frame and Figure 1b shows the time sequence of frame transmissions and acknowledgements. In the discussion below we will assume an 8-tone system, so each frame transmission consists of 64 subpackets. The sender terminal buffers its message subpackets and sends these according to their sequence number. Subpackets that are detected in error at the acceptor terminal are immediately retransmitted in the next frame. We now discuss a number of possible error-control strategies.

In standard ARQ the checksum of each subpacket is computed and subpackets found in error are discarded. A subpacket is retransmitted until an error-free version is delivered to the receiver. A flow chart for this scheme is shown in Figure 2. This scheme is simple to implement and will perform well as long as the subpacket error rate is low. The system completely breaks down

8 fsk tones
8 subpackets per tone
144 bits per subpacket

	1	2	3	4	5	6	7	8
fsk#1								
fsk#2	H							
fsk#3	E							
fsk#4	A							
fsk#5	D							
fsk#6	E							
fsk#7	R							
fsk#8								

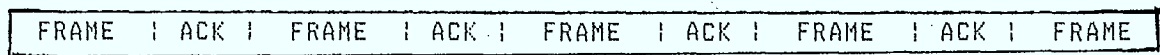


Figure 1. Frame Structure and Frame Transmission Sequence

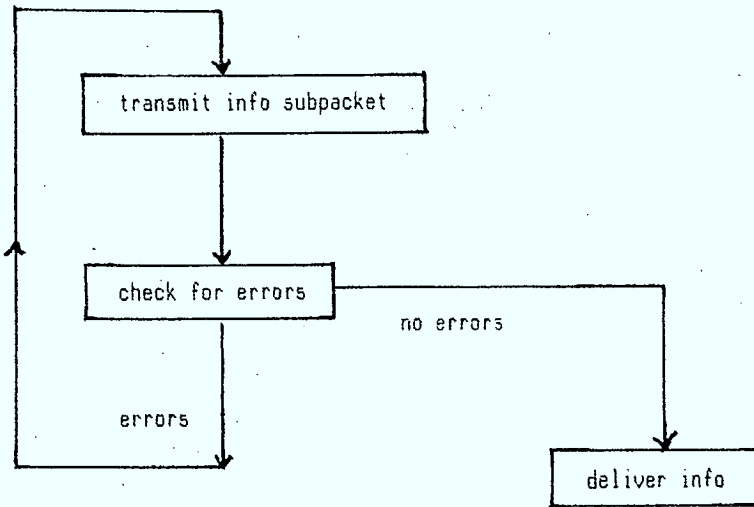


Figure 2. Standard ARQ

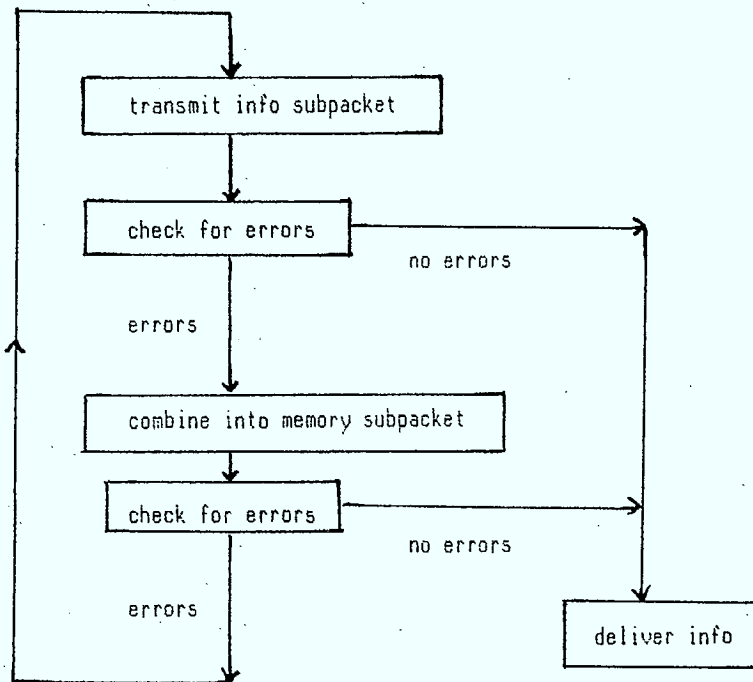


Figure 3. Memory ARQ without Coding

however as soon as the subpacket error rate is sufficiently high that virtually all subpackets contain errors.

Standard ARQ can be viewed as time-diversity transmission without combining. It is then apparent that the threshold where the throughput drops to zero can be extended through the use of combining: an A/D converter is introduced at the modem output and a soft version of the subpacket is passed to the decision device. The device obtains a hard-decision version of the subpacket and computes the checksum as in standard ARQ. When errors are introduced, however, the soft version of the subpacket is stored for possible combining with subsequent retransmissions of the subpacket. We will refer to this approach as memory ARQ without coding. Figure 3 gives a flow chart for this approach.

Lin and co-workers have developed a family of hybrid ARQ error correction/detection protocols based on the observation that in standard ARQ a subpacket and its subsequent retransmission form a simple rate 1/2 repetition code. The performance of the system can therefore be improved by using more powerful rate 1/2 error correcting codes. Each "subpacket" now has two versions: the first consists of the information followed by CRC bits; the second consists of a coded version of the information followed by CRC bits. The code used is invertible in the sense that the information can be recovered unambiguously from the coded subpacket. By alternating the transmission of the two subpackets, it becomes possible to attempt a recovery of the information through error correction. Figure 4 gives a flowchart for this scheme. Note that in this approach subpackets are discarded after they have participated in an erroneous decoding.

Memory can be introduced into the hybrid ARQ schemes by not discarding the subpackets after unsuccessful decodings, but instead keeping them for possible combining with subsequent retransmissions. The combining should result in a reduction in the number of retransmissions under poor channel conditions. We will also refer to this approach as memory ARQ with coding. A separate memory version of the information and parity subpackets is necessary as shown in the flowchart in Figure 5. Note that the approach naturally leads to the use of soft-decision decoding. The principal objective of the project was to evaluate the performance of various codes when combined with this approach.

In the next section we will describe the design and setup of an experiment that was carried out to experimentally compare the performance of these basic ARQ schemes when applied to data transmission over HF radio. In a later section we will consider modifications of the basic schemes.

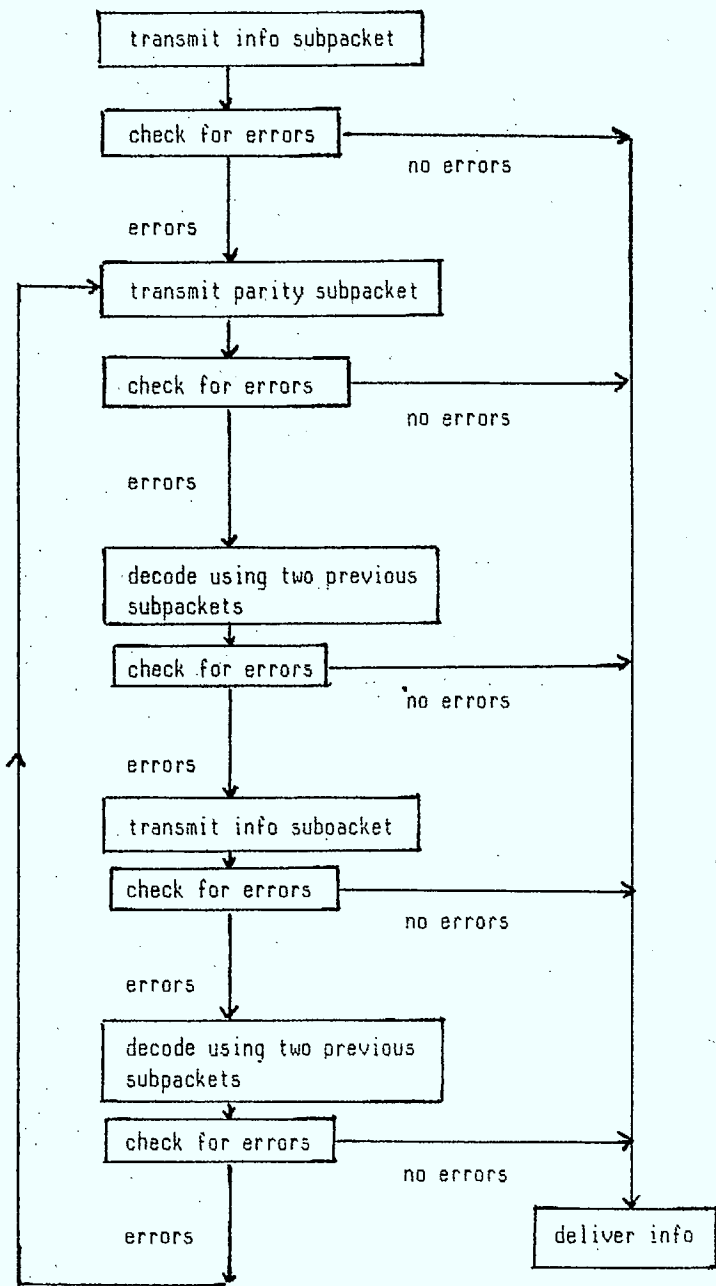


Figure 4. Hybrid ARQ

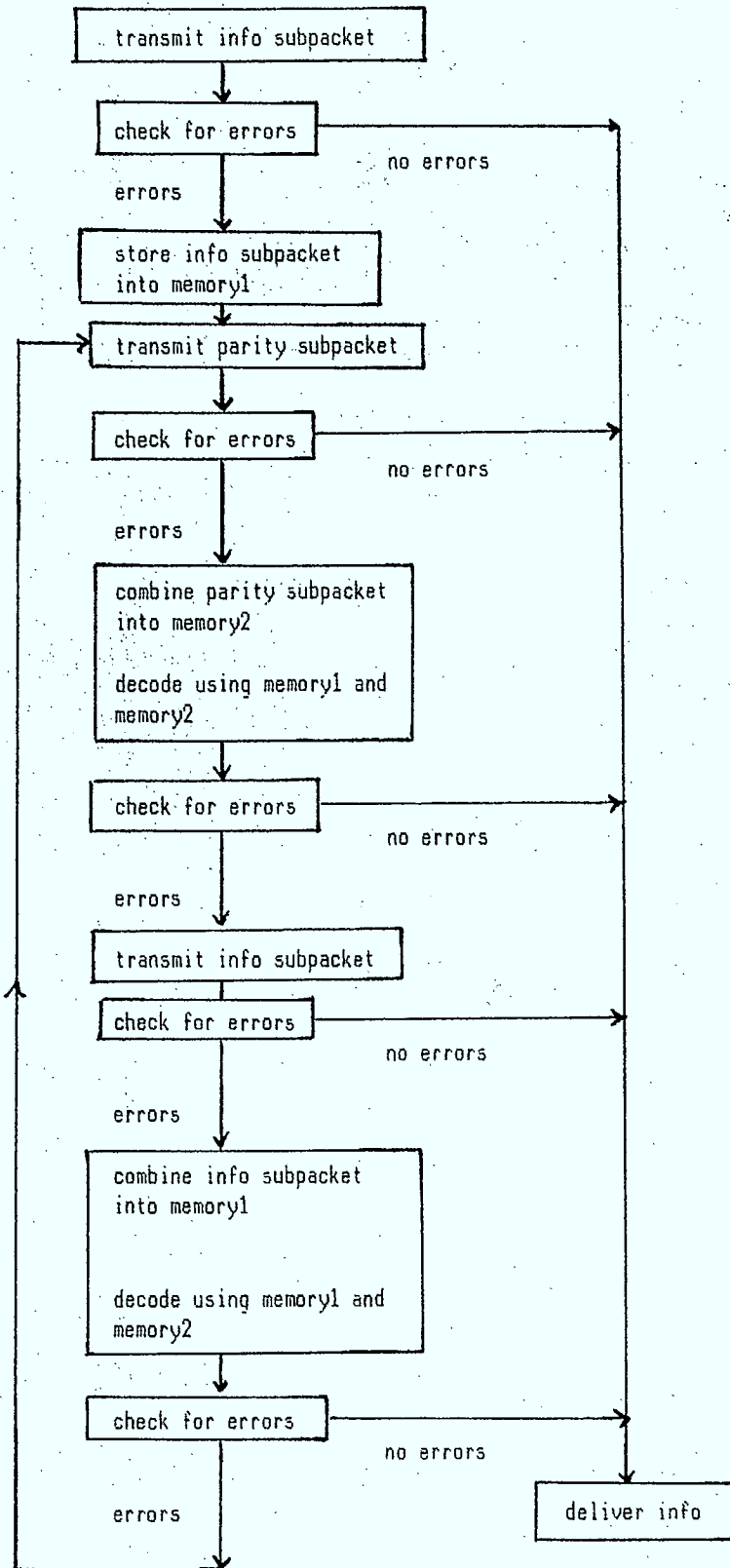


Figure 5. Memory ARQ with Coding

3. EXPERIMENT DESIGN AND SETUP

In this project it is assumed that the terminal is implemented around a personal computer. It is also assumed that the error transmission protocol is to be implemented using the pc. The selection of codes tested is therefore dictated by their feasibility for implementation in software. In previous phases of the project it had been determined that short block codes as well as short constraint-length convolutional codes are feasible for real-time or near real-time soft-decision decoding. The following codes were therefore selected for testing:

1. (8,4) Reed-Muller block code
2. (16,8) block code obtained by shortening the (17,9) cyclic code with generating polynomial $1 + x^3 + x^4 + x^5 + x^8$
3. 35/23 constraint length 5 convolutional code
4. 171/133 constraint length 7 convolutional code.

In order to compare the relative effectiveness of the codes it is necessary that they undergo the same channel transmission conditions. This can be accomplished in a multi-tone system by transmitting the different encoded sequences simultaneously over the available tones. A data gathering experiment was carried out in which the frame shown in Figure 6 was continuously transmitted. The frame length was selected to correspond to that of the two-tone system. A 241 symbol synchronization sequence followed by a 16 symbol frame number was sent instead of the header. Eight subpackets of length 144 symbols followed. The 1152 bit information sequence was obtained from a pn sequence generator. The block code parity sequences were obtained by segmenting the information sequences into groups of 4 or 8 symbols and then encoding using the (8,4) and (16,8) codes respectively.

The convolutional code sequences were obtained in a slightly different manner. The information sequences were divided into 8 subsequences of 144 bits each. The last 4 or 6 bits of the sequences were set to zero. The resulting sequences were then convolutionally encoded using the 35/23 and 171/133 codes. The resulting coded sequences have the property that the state of the encoder begins and ends in the zero state.

The tone assignments to each sequence were made so that the sequences involved in the same decoding would not be in adjacent tones. To control for differences in the conditions in the various tones, the transmitted frame was cyclically shifted by one tone so that in effect the transmitted pattern consisted of eight rotated versions of the same fixed-frame.

The transmission of the above frame allows us to evaluate the performance of the various ARQ schemes under identical channel transmission conditions. If a given subpacket is found in error, its "retransmission" is automatically found in the next frame. By

8 information and coded sequences

Each sequence consisting of 8 subpackets

Each sequence transmitted on one tone for the duration of a frame

Sequence tone assignments rotated by one position after each frame transmission

			1	2	3	4	5	6	7	8
cc23										
cc133	S	N								
info	Y	U								
bc16	N	M								
cc35	C	B								
cc171	H	E								
bc8		R								
info										

Figure 6 Experiment Frame Structure

design the temporal separation between these subpackets corresponds closely to that encountered in an actual system. Thus the performance results obtained should be indicative of the performance in a fully operational system, as well as valid for comparing the relative performance of the various schemes. The frame design also allows us to evaluate frequency diversity and coded frequency diversity performance by decoding appropriate pairs of subpackets.

Appendix A in the Interim report describes IBM PC compatible boards that were designed for the transmitter and receiver. The transmitter board accepts data from the pc and drives an 8 tone modem. The receiver board accepts 8 "eye" (baseband digital) signals and a recovered clock from the modem and then samples each "eye" a multiple number of times. These samples are DMA'ed to memory and later processed to obtain an approximation to an integrate-and-dump, four-bit, soft-decision-detected sequence. In the transmission tests the detected sequence were segmented into blocks that corresponded to a complete frame transmission which were stored in disc on an IBM PC/XT. The experimental data was later transferred to floppy discs and shipped to the University of Toronto where the performance measurements were made using an HP9000 computer system.

The segmentation of the received sequences into blocks was not synchronized to the frame boundaries. A preliminary search of the frame synch section was first made on an IBM PC as follows. A binary template of the first 32 bits of the synch sequence was shifted and compared to received sequences by taking the most significant bit of the sequences and adding the number of matches. This simple technique was successful in locating the frame boundary for virtually every frame. It was found that the recorded sequences contained a significant number of bit slips which would probably make correlation over the entire frame ineffective.

The original unsynchronized experiment files were transferred from the IBM PC to the HP9000. Prior to the transfer a modulo 256 sum was made of the characters in each block to obtain a parity check that was later used to verify that the files were not altered in the transfer. The location of the first subpacket in each frame was then carried out by correlating a template of the first subpacket against the eight rotated versions of the received sequence. The correlation computation used all 4 bits in the received symbols. The search for the boundary was limited to the vicinity of the location predicted by the first set of measurements that had been done on the PC. Once a decision on the location of the first subpacket was made, it was assumed that the following symbols corresponded to the rest of the subpackets. The entire frame was then unrotated and a file containing unrotated subpacket sequences was then produced. This file was the input to the programs that carried out the performance evaluation of the various protocols.

4. RESULTS

RAW DATA

The experimental data files are stored as character arrays. In order to produce an output "map" that would allow us to easily peruse the transmission data, a program was written that compared each subpacket with a corresponding template. The number of hard errors were counted and a file indicating the number of errors in each subpacket was produced. A sample of the output of such a program is shown in the next page. Each line in the listing corresponds to a frame or packet which consists of 64 subpackets. The frames (packets) in each experiment are numbered from zero onwards. Each character in the line indicates the number of errors in the corresponding subpacket. The characters are arranged in groups of eight according to the order of transmission. Thus the first 8 characters correspond to the first cc23, cc133, info, bc16, cc35, cc171, bc8, and info subpackets that were transmitted in parallel, the second group of 8 characters correspond to the next 8 subpackets, and so on. The characters values correspond to the following range of error counts:

character	error count range
0-9	0-9
a-j	10-19
@	20-29
#	30-39
\$	40-49
%	50-59
^	60-69
&	70-79
*	80-89
(90-99
A	100-109
B	110-119
C	120-129
D	130-139
E	140-149

Note that the alphanumeric characters are the uppercase characters corresponding to matching numbers on the keyboard.

The 5-digit number on the right hand side of the frame is the correlation value of the block of 144 bauds that best matched the template for the first subpacket of the frame. There are 144 bauds with 8 symbols per baud and 4 bits per symbol. The maximum score is for a complete match and is thus therefore $144 \times 8 \times 15 = 17280$; the score for a completely uncorrelated sequence would be $17280/2 = 8640$.

A total of 12 experiments were carried out in the two-week period beginning on September 12, 1984. The last three experiments were used in obtaining the performance results presented below. The "maps" for these experiments are included in

;H2J
data experimentfile [/users/rose/data/realexpt.001]: ;H2J system main menu:

- b = browse through file for hard errors
- h = histogram of experiment channel
- r = browse through file for hard errors (raw)
- q = quit

option [b]?:

channels are: cc23 ccl33 info bcl6 cc35 ccl71 bc8 info

i am looking at experiment file: realexpt.11

pkt #	021.2	1....1.	1..1.g.0	...6a.a	.af.03.5	9#0\$99b7	&&&&^&&&	&&^^&(^	(15894) (...*....)
pkt #	1	6.7e.251	9..1.a61	..0.1...	c....4a5	j....00.	j....9h.	8....b63	52....2	(15695) (...**....)
pkt #	2	3b....88	3b....83	.41..1.2	.5....21	14....21	.3.....	.5....14	11....11	(15797) (...**....)
pkt #	3	615....b	633....6	348....4	..2....	a.c....	g.d....4	1.3....1id..	(15551) (...**....)
pkt #	4	9e.f7...	80501..1	27.7....	166d....	361d....	3213....	cb2b....	.4.7....	(15386) (...**....)
pkt #	5	.d114..2	.1.1.2..	.4e1j...4...	..1123...4...	5.4.6.5.	.491i...	(15731) (...**....)
pkt #	6	..3615..	..14.5..	..7528..2...	..38.8..	..33.7..	..617..	..1111..	(15824) (...**....)
pkt #	7	...17.d.	...46.e.	...92.7.	...91.7.	1..31.6.	c652i2e1497e	%^^%^^%	(15725) (...**....)
pkt #	82.8	...18.7	%^^%^^%	&&&&&&^	&^&&&&&&	&^&&&&&&	&&^^(&&	&&&&&&^	(15841) (...**....)
pkt #	9	3....11.	4....11	3....622	.2....1.	3.....	1.....	6....22	0....4c1	(16027) (...**....)
pkt #	10	38....25	15....54	.4.....	33....34	12....2.	13.....	.f....4c	2f....d7	(15854) (...**....)
pkt #	11	1.1....	948....2	d.a....	j.h....i	2.3....4	1.6....4	414....2	d1i....	(16034) (...**....)
pkt #	12	f114....	b417....	21.2....	..13....	2124....	.1.1....	5e3j.2..	a710....	(15849) (...**....)
pkt #	13	..1.3...	..1519...	..15.5...	..434...	..2716...	..5a1b...	..8915...4...	(15919) (...**....)
pkt #	1411312.1	..20204.	..e8.g2.	..08.c..	..f8.8..	..3.22..	(16097) (...**....)
pkt #	15	..4324.	..15.5.	48.5..3.	232.1i.2	..j0200	..44.9.	..j51a.	...0..6.	(15952) (...**....)
pkt #	16	.45...12	.cg1..1.	.#ei....	.2.120.0	.50.54.6	.g525...	.42#91..	1..41j38	(15838) (...**....)
pkt #	17	c....9.	a....7.	8....74.	4....42.	5....752	8....511	5.80.6..	2.#5.7.1	(15732) (...**....)
pkt #	18	3i....7a	1a....64	1a....7b	1b....a6	.1...1.	.b....6	.5....32	13....14	(15239) (...**....)
pkt #	19	8.d....2	3.3....8	128....b	1434....4	360f1b12	735....	..3....	aig....	(15700) (...**....)
pkt #	20	12.9....	5b36....	2..4....	1115....	51.d....	3121....	.424....	.4.4....	(15901) (...**....)
pkt #	21	.5..2...2..	.44.8...	.41.7...	...2...	.12.5...	..1.1...	..4.5...	(15825) (...**....)
pkt #	22	..5.3..	..7a2d1.	..1335..	..32.3..	..941b..	..2..3..	..22fa06.	..1344#62	(15932) (...**....)
pkt #	23	..9g7f2	..d.161111.	...3f2i.	...64.b.	...21.2.	...5637.2.	(15339) (...**....)
pkt #	24	1...23.2	1...212.	2....52	3...12.3	...241b5.413.647.d	(15800) (...**....)
pkt #	25	3....1.	5....111	8....26.	7....16.	b....382	4....1..	f....1.	6....722	(16003) (...**....)
pkt #	26	131....2	3c%...1	.hb...c7	38...1.2	.2...2.	7e1...9h	22....11	.b....3a	(15492) (...**....)
pkt #	27	134.15.4	..5....	2.5....2	2.3....3	..8....1	31d1...1	..4....	325....a	(15769) (...**....)
pkt #	28	2626....	1..3....	1625....	171f....	a31a....	ad90...5	364e....	9625...2	(15790) (...**....)
pkt #	29	.b316...	31472.1.	.48.7...	.2126...	..218...	.1439...	.25.5...	.34.3...	(15747) (...**....)
pkt #	30	00e0fidj	&&&&&&^	^&&&&&&^	^(^&&&&&&	&^&&&&&&	#^(&&&&&&	(&^^(&&&	&&^^&&&^	(13736) (...**....)
pkt #	31	...eb.f.	...3..72	...fg3a.	..2j70fh	113ce.4.	ic25.31.	7..ce3d.	...224.	(15638) (...**....)
pkt #	32	...1315212	#.1...3.	5...1106	.131..3.	...23.91.12	.1...a.e	(15811) (...**....)
pkt #	33	5.....	^%^^%^^%	(&&&&&&	(&&&&&&	^^&&&&&&	&^&&&&&&	(&^&&&&&	&&&&&&^	(15937) (...**....)
pkt #	34	3d...179	.2....2.	.1....11	19....5	1h....bi	.6....a1	.5....1	14.71..1	(15703) (...**....)
pkt #	35	5.4....5	216....4	316....	c.j....5	2.81...6	119....1	i1c....9	7.#...#	(15844) (...**....)
pkt #	36	51.7....	3318....	2124...1	43.9....	2428....	152a....	.1.8....	32c7....	(15866) (...**....)
pkt #	37	...12...	41423.6.	11518.g1	.77.f...	.ed3i...	..1.4...	.2339...	...12...	(15833) (...**....)
pkt #	38	..2413..	..221b..	..544fa.	..905e9.	g.401052	..bh1j..	..88.9..	..7.11..	(15901) (...**....)
pkt #	39	...3c.g.	...7110.	...1214.	...c1512	1...0d#	16.18004	..8549.	...5g1c1	(15712) (...**....)
pkt #	40	.1..23.73	...2017	...31.8	...1.11	1...1.8	#....c2#	2....i2a	(15801) (...**....)

the Appendix . It can be seen in these maps that the frame recovery was successfully carried out in all three of these experiments.

The sample output shows a "barber-pole" effect that was found in all of the experiments. This indicates that certain tones consistently exhibited worse error rates than other tones. This is probably due to dc offsets in the eye signals. The sample output also shows what appear to be bit slips. In packets number 8 and 33 it can be seen that the error rate in the subpackets becomes approximately 50% from some point in the frame onwards.

PERFORMANCE OF MEMORY ARQ PROTOCOLS

Computer programs were written in Fortran to implement the standard ARQ and memory ARQ protocols. The programs computed the distribution and the mean of the number of transmissions per delivered subpacket and the relative frequency of the points in the algorithms where an errorfree packet is delivered. Because of the non-stationary nature of the HF channel, these statistics were computed separately for blocks of 50 frames which correspond to periods of slightly less than 12 minutes.

Tables 1-3 show the results for experiments #10 through #12 respectively. The Tables show the mean number of transmissions per delivered subpacket and the total number of subpackets delivered during a block.

The map for experiment #10 in the Appendix shows that transmission conditions were quite good during this experiment. Most of the subpacket transmissions were received errorfree after the first transmission. As well, when a retransmission is required, it is very likely that it will be received errorfree. The specific features of the various ARQ schemes are therefore seldom required, and consequently the performance of the schemes are nearly identical.

Experiment #11 exhibits transmission conditions quite worse than those in the previous experiment. Furthermore, the conditions get worse as the experiment progresses. This can be seen in the corresponding map and is clearly evident in the statistics shown in Table 2. Differences in the different ARQ schemes are apparent:

--the use of combining in memory ARQ without coding makes it uniformly better than standard ARQ;

--the use of coding results in further improvements;

--the difference in error-correcting power of the codes becomes apparent only as the channel conditions worsen; Thus the constraint length 7 convolutional code, while always the best, only becomes significantly better as the channel conditions become very poor. The weakest of the codes (bc8) performs almost as well

TABLE 1. RESULTS FOR EXPERIMENT #10

mean number of transmissions per delivered subpacket

block #	std	memory	bc8	bc16	cc5	cc7
1	1.42	1.40	1.39	1.39	1.39	1.38
2	1.06	1.06	1.06	1.05	1.04	1.03
3	1.11	1.11	1.11	1.08	1.12	1.09
4	1.14	1.14	1.14	1.15	1.15	1.18
5	1.26	1.24	1.24	1.24	1.21	1.27

delivered subpackets per block

block #	std	memory	bc8	bc16	cc5	cc7
1	356	361	361	363	364	366
2	397	397	397	401	400	401
3	401	401	401	401	397	401
4	399	401	401	401	398	401
5	210	212	212	212	212	211

TABLE 2. RESULTS FOR EXPERIMENT #11

mean number of transmissions per delivered subpacket

block #	std	memory	bc8	bc16	cc5	cc7
1	1.88	1.75	1.71	1.56	1.58	1.52
2	1.62	1.55	1.50	1.50	1.39	1.47
3	1.71	1.59	1.55	1.50	1.54	1.47
4	2.06	1.86	1.68	1.71	1.65	1.64
5	2.57	2.10	1.87	1.86	1.84	1.80
6	2.41	1.99	1.83	1.73	1.68	1.67
7	2.29	2.02	1.88	1.88	1.68	1.82
8	2.15	1.84	1.71	1.67	1.68	1.63
9	3.23	2.76	2.30	2.22	2.16	2.03
10	3.54	3.20	2.75	2.54	2.49	2.42
11	6.70	5.93	5.38	5.01	4.92	4.58
12	---	---	---	---	57.36	13.18
13	11.04	10.71	10.28	9.72	8.75	8.53

delivered subpackets per block

block #	std	memory	bc8	bc16	cc5	cc7
1	324	346	352	377	365	374
2	358	373	385	391	399	397
3	348	373	383	391	388	396
4	300	328	362	370	369	373
5	256	308	347	361	356	360
6	272	323	351	363	375	394
7	283	315	339	334	348	349
8	298	342	369	371	385	391
9	211	242	290	305	311	321
10	194	214	245	272	279	286
11	112	126	139	148	152	163
12	0	0	1	3	13	60
13	67	69	72	75	84	86

TABLE 3. RESULTS FOR EXPERIMENT #12

mean number of transmissions per delivered subpacket

block #	std	mem	bc8	bc16	cc5	cc7
1	1.42	1.35	1.35	1.33	1.42	1.38
2	1.69	1.56	1.52	1.52	1.47	1.51
3	1.48	1.42	1.39	1.38	1.36	1.40
4	3.59	3.33	3.15	3.19	2.96	3.01
5	1.86	1.82	1.76	1.64	1.69	1.72
6	14.14	14.14	8.83	15.84	9.17	5.90

delivered subpackets per block

block #	std	mem	bc8	bc16	cc5	cc7
1	375	389	391	395	389	391
2	340	365	377	367	383	373
3	363	377	385	389	394	391
4	190	203	215	212	228	227
5	326	332	342	344	355	351
6	42	42	66	38	64	100

as the other codes in most of the blocks.

In experiment #12 the channel conditions are intermediate between those of the two previous experiments. Again the performance of the various schemes are essentially the same except in the last block where channel conditions are very poor.

5. OTHER ARQ PROTOCOLS

In this section we will describe a number of protocols that were devised to improve on the performance of the basic memory ARQ protocols already discussed above. Section 6 will present the corresponding performance results.

All ARQ schemes automatically adapt to changing channel conditions through the automatic retransmission of erroneous subpackets. For the multi-tone multiple-packet per-frame system under consideration the penalty incurred in gaining this adaptivity is increased transmission delay as well increased complexity in buffer management and sequence numbering operations. Thus it is desirable if the protocol operates so that the number of retransmissions is kept low, while maintaining the throughput as high as possible. For example, if the average number of transmissions per subpacket is around 2, it may be preferable to concede some throughput performance by transmitting both versions of the subpacket in the same frame the first time that they are transmitted. Subsequent retransmissions would follow the usual protocol. This type of adaptive arq protocol requires that the channel condition be tracked. This can be done automatically by observing the rate at which retransmission requests are being made.

In order to evaluate the performance of the above type of adaptive algorithm, a computer program was written in which dual coded frequency diversity is used the first time that a subpacket is transmitted. If either of these subpackets is received error free, then the information can be recovered immediately. If both are received in error, then a soft-decision decoding can be carried out. Further retransmissions are necessary only if this decoding is unsuccessful. We note that subsequent retransmissions do not use frequency diversity, but instead follow the memory ARQ protocol.

A close examination of the memory ARQ protocol discussed in the previous sections reveals that there are more than one way in which a "memory" algorithm can operate. In the "type 1" memory algorithm already discussed, a subpacket with detected errors is immediately combined into the corresponding memory subpacket prior to the soft-decision decoding step. In what we will define as the "type 2" memory ARQ algorithm, a subpacket found in error is not immediately combined into the memory subpacket. Instead the subpacket is used with the complementary memory subpacket in a soft decision decoding step. Only when this decoding fails is the subpacket combined with the memory subpacket. It can be argued

that the type 2 algorithm should perform better than the type 1 since a subpacket found with errors is less "suspect" than one that has already participated in an unsuccessful decoding attempt.

The type 2 memory ARQ protocol and the Lin protocols can be viewed as two extremes of a class of protocol in which each subpacket can be either discarded or combined after each decoding step. In the Lin protocol the oldest subpacket is always discarded after it has participated in a decoding failure. In the type 2 memory ARQ protocol, subpackets are always combined into the corresponding memory subpacket. A natural middle ground to these two extremes is a protocol in which a subpacket is either discarded or combined into memory depending on its potential usefulness in subsequent decoding steps. The use of soft decision decoding provides us with the natural measure of a subpacket's potential usefulness, namely, the metric that is associated with each decoding. If the metric is less than some threshold then the subpacket is combined into memory; if the metric exceeds the threshold then the subpacket is discarded. The value of the threshold must be optimized to yield the maximum improvement in performance. Some results on this method which we will dub memory ARQ with selective discard, will be presented in the next section.

6. PERFORMANCE OF OTHER ARQ PROTOCOLS

The experimental data used in evaluating the performance of memory ARQ protocols can also be used to evaluate the performance of the protocols discussed in Section 5. Because of their relatively high error rates, experiments #11 and #12 were used to obtain performance results. The (8,4) code and the constraint length 7 code were used because they were the extreme cases of the codes considered in terms of error correcting capability and implementation complexity. The results are given in Tables 4 through 7.

A comparison of the first two columns in all the tables shows that the type 2 memory ARQ protocol is always better than the type 1 memory ARQ protocol. For most channel conditions the two schemes have nearly the same performance, but under very poor channel conditions the type 2 protocol is clearly better.

A comparison of the third and fourth columns shows that the Lin hybrid ARQ schemes do extremely well given the fact that they do not use combining. The Lin protocol with hard decision decoding nearly equals the performance of the Lin protocol with soft decision decoding except under very poor channel conditions. The Lin algorithm with soft decision decoding and the type 2 memory ARQ algorithm have very nearly the same performance.

The last two columns give the performance of the memory ARQ protocol with selective discard. Two values of threshold were used 500 and 1000 (with 4 bit quantization, the maximum distance

TABLE 4. PERFORMANCE OF OTHER ARQ SCHEMES
convolutional code of constraint length 7
experiment #11

mean number of transmissions per delivered subpacket

block no.	memory type1	memory type2	lin hard	lin soft	adapt memory type1	adapt lin soft	memory thresh 500	memory thresh 1000
1	1.51	1.50	1.54	1.50	1.10 2.10	1.09 2.09	1.50	1.50
2	1.47	1.47	1.51	1.47	1.01 2.01	1.01 2.01	1.47	1.47
3	1.47	1.47	1.49	1.47	1.01 2.01	1.01 2.01	1.47	1.47
4	1.63	1.62	1.67	1.62	1.12 2.12	1.10 2.10	1.62	1.62
5	1.76	1.75	1.85	1.75	1.11 2.14	1.10 2.12	1.75	1.75
6	1.67	1.66	1.82	1.66	1.08 2.08	1.06 2.06	1.66	1.66
7	1.83	1.81	1.92	1.82	1.15 2.15	1.13 2.13	1.82	1.82
8	1.63	1.63	1.71	1.63	1.03 2.03	1.03 2.03	1.62	1.62
9	2.01	1.96	2.10	1.96	1.35 2.35	1.27 2.27	1.96	1.96
10	2.40	2.31	2.96	2.32	1.59 2.60	1.54 2.53	2.32	2.32
11	2.27	2.24	3.08	2.29	1.50 2.53	1.50 2.53	2.29	2.22
12	4.75	3.87	----	3.68	3.85 4.92	2.88 3.90	3.68	3.68
13	2.91	2.84	3.40	2.86	1.85 2.86	1.85 2.86	2.86	2.86

TABLE 5. PERFORMANCE OF OTHER ARQ SCHEMES
 (8,4) block code
 experiment #11

mean number of transmission per delivered subpacket

block no.	memory type1	memory type2	lin hard	lin soft	adapt memory type1	adapt lin soft	memory thresh 500	memory thresh 1000
1	1.72	1.69	1.92	1.69	1.11	1.10	1.69	1.69
					2.11	2.10		
2	1.48	1.48	1.59	1.40	1.03	1.03	1.48	1.48
					2.03	2.03		
3	1.54	1.54	1.67	1.53	1.02	1.02	1.54	1.54
					2.02	2.02		
4	1.68	1.67	1.83	1.67	1.10	1.10	1.68	1.68
					2.10	2.10		
5	1.83	1.83	2.16	1.83	1.16	1.16	1.83	1.83
					2.19	2.19		
6	1.82	1.80	2.02	1.80	1.12	1.11	1.80	1.80
					2.12	2.11		
7	1.86	1.85	2.02	1.80	1.12	1.26	1.83	1.83
					2.12	2.26		
8	1.70	1.70	1.98	1.71	1.10	1.10	1.71	1.71
					2.11	2.11		
9	2.28	2.19	2.69	2.21	1.47	1.42	2.21	2.21
					2.50	2.45		
10	2.72	2.61	3.30	2.56	1.89	1.83	2.58	2.56
					2.92	2.86		
11	2.95	2.95	3.69	2.94	1.85	1.81	2.94	2.94
					2.89	2.84		
12	----	----	----	----	----	----	----	----
13	3.55	3.54	3.91	3.50	2.06	2.06	3.49	3.50
					3.09	3.09		

TABLE 6. PERFORMANCE OF OTHER ARQ SCHEMES
 (8,4) block code
 experiment #12

mean number of transmission per delivered subpacket

block no.	memory type1	memory type2	lin hard	lin soft	adapt memory type1	adapt lin soft	memory thresh 500	memory thresh 1000
1	1.36	1.36	1.40	1.36	1.02 2.02	1.02 2.02	1.36	1.36
2	1.52	1.51	1.60	1.51	1.15 2.15	1.14 2.14	1.51	1.51
3	1.38	1.38	1.42	1.38	1.06 2.06	1.05 2.05	1.38	1.38
4	2.07	2.00	2.15	1.98	1.53 2.53	1.45 2.46	1.98	1.98
5	1.57	1.57	1.71	1.51	1.37 2.37	1.30 2.30	1.98	1.98
6	8.86	7.07	13.3	7.90	6.12 7.18	5.47 6.52	7.90	6.49

TABLE 7. PERFORMANCE OF OTHER ARQ SCHEMES
convolutional code of constraint length 7
experiment #12

mean number of transmission per delivered subpacket

block no.	memory type1	memory type2	lin hard	lin soft	adapt memory type1	adapt lin soft	memory thresh 500	memory thresh 1000
1	1.38	1.38	1.39	1.38	1.02	1.01	1.38	1.38
2	1.49	1.49	1.52	1.49	2.02	2.01	1.49	1.49
					2.10	2.09		
3	1.40	1.40	1.44	1.40	1.04	1.04	1.40	1.40
					2.04	2.04		
4	2.01	1.91	2.02	1.91	1.45	1.39	1.91	1.91
					2.45	2.39		
5	1.57	1.55	1.60	1.55	1.17	1.15	1.55	1.55
					2.17	2.15		
6	5.89	4.48	14.9	4.58	3.85	2.81	4.58	4.53
					4.89	3.84		

between two 144 symbol sequences is $144 \times 15 = 2160$). It is clear that no significant gain over type 2 memory ARQ or the Lin protocols are obtained.

A comparison of the performance of the (8,4) code and the constraint length 7 convolutional code shows that the (8,4) code nearly always performs as well as the more complex convolutional code. The only exceptions are block #6 in experiment #12 and block #12 in experiment #11. The dashed lines in the table indicate that not enough subpackets were delivered in the block to obtain a reliable estimate. Thus it appears that opting for the simpler code entails no significant loss in performance over the range of reasonable channel transmission conditions.

Finally columns five and six show the performance of the adaptive algorithm if it were operating in the dual-diversity mode for each first subpacket transmission. The entry for each block is the mean number of frame time delays before a subpacket is delivered errorfree to the receiver. The second entry is the total number of subpackets required to deliver the errorfree subpacket. The first entry is the subpacket delay and the reciprocal of the second entry is the subpacket throughput efficiency. It can be seen that the algorithm in adaptive mode will indeed reduce the delay, and that it will do so without a significant penalty in throughput.

7. CONCLUSIONS

The performance of a number of ARQ protocols has been evaluated over channel conditions ranging from quite good to very poor. This performance evaluation is highly computation intensive, so an exhaustive study using a larger set of experimental data is not feasible. However the following conclusions can be made with some confidence. First, the use of coding instead of simple combining is beneficial in that it extends the threshold of channel conditions in which reasonable throughputs are attainable. Second, increasing the complexity of the code results in significant performance improvement only under severe channel transmission conditions. Third, the use of memory combining with coding does not significantly improve the performance over the simpler indiscriminate subpacket-discard policy of the Lin hybrid protocol. Fourth, the subpacket delay can be kept low by switching to dual coded frequency diversity transmission when the channel conditions deteriorate.

APPENDIX

ARQ experimentfile [/users/rose/data/realexpt.001?]: system main menu:

- a = analysis experiment data
- b = browse through experiment for hard errors
- h = histogram of experiment channel
- n = next experiment file
- q = quit

option [a]?:

channels are: cc23 cc133 info bc16 cc35 cc171 bc8 info

i am looking at experiment file: /users/rose/data/realexpt.10

pkt #	0	^^^&&^&	^^&&&&&	*&^^&^^	^^&^^&^^	&&&^^^&	^^&^^&^^	&&^^&^^	&&&&&&^&	(9152) (.....)
pkt #	1	\$\$\$\$\$\$\$	&&&^&^&	fafgg@ae1.....	(16595) (.....)
* pkt #	2	.1.....2	.2.....	%%\$\$\$\$%	edicdfjf	^^&^^&^^	&&*&&&&&	^^&^^&^^	&&^&^&^&	(16439) (.....)
pkt #	3	.1.....2	.2.....	%%\$\$\$\$%	&^&&^&&&	^^&^^&^^	&*&&&&&&	^^&^^&^^	&^&^&^&	(16439) (.....)
* pkt #	4	.1.....2	.2.....	%%\$\$\$\$%	&&&^&^&	&^&&^&*	&^&&&^&*	^^&^^&^^	&^&^&^&	(16439) (.....)
* pkt #	59.1.1..	#####	(16450) (.....)
pkt #	6	^^&^^&^^	^^&^^&^^	&*&&&&&&	^^&^^&^^	&^&^&^&	(16450) (.....)
* pkt #	7	1.....11	%%\$\$\$\$%	i@e@fhg@1a.3.8.	(16563) (.....)
pkt #	8	1.....	1.....	(16435) (.....)
pkt #	92.....	2.....1..	Y.....2..	(16669) (.....)
pkt #	102.....	.11.....2.....	(16557) (.....)
pkt #	111	...1.1.5	...1....	.4...a41	(16466) (.....)
pkt #	12	.2.41...	...1....	1.....	1..1....	(16389) (.....)
pkt #	13	..4.63.1	4..141.1	..3.71..	.1..1..	6d@6@4b3	.4..1..2	@cg@dj@#	(16140) (.....)
pkt #	14	6.41113.	..1.....	.12...11	53b21a.61..	c5@e7@64	..1.....	...2....	(16183) (.....)
pkt #	15	.3.2..2.	.5.@1.1.	6...2.27	..8....	b5g185@8	1.319c@a	.a.71.1.	(16295) (.....)
pkt #	16	1b16..4.	d1.41q2@1	129.b21234	e2669282	(16137) (.....)
pkt #	17	1.....	121114..	1...1..1..	b.....5.	111.3..1	e4...132	(16540) (.....)
pkt #	18	499..1721..	.1...1.	121.....	.2.....2.1.	(15970) (.....)
pkt #	19	.a.d116.	^^^&&^&	%&&&&&^&	&&^^&&&^&	&&^&^&^^	&^&&&&*&	&&&&&&^&	&^&^&&&&	(16025) (.....)
pkt #	20	f15.38579.8....	.9.b...1	..4.46..	dh@f##^#	&^^^&&&	&*&&&&&	(15847) (.....)
pkt #	21	.7.52..5	d6.6.2i.	%%##@##	&^^*##&&	&^^^&&&	^&^^*##&	^^&^^&&&	^&^&^&^&	(15990) (.....)
pkt #	22	.c1j6h882..	2.1.....	..631d1.2..	(15418) (.....)
pkt #	231....1.	(16574) (.....)
pkt #	24	(16614) (.....)
pkt #	25	2.....	(16537) (.....)
pkt #	261.....	.1.....	(16711) (.....)
pkt #	271.....	...3.1	...1...	...1...	(16602) (.....)
pkt #	281....7..	..b.e@@6	..613...	(16501) (.....)
pkt #	291..	(16546) (.....)
pkt #	301....	...1.1..	(16574) (.....)
pkt #	31	c#db9j7@	^^&&*&&&	&&&%&^&	*&^&&^^	(16605) (.....)
pkt #	3221	2..5.1394.4	(16453) (.....)
pkt #	331..	2.....	611...2111..	..1...1.	53211..1	8b61c.26	(16477) (.....)
pkt #	34	54711114	69e8@bh.	@hg@#f@9	442..111	58g74889	.1.1....	(15912) (.....)
pkt #	3521....	e.e....	..2.....1	(16539) (.....)
pkt #	362....	(16428) (.....)
pkt #	37	11.....	..1.1..5..1..1...	(16494) (.....)
pkt #	383..1..	.6221143	(16543) (.....)
pkt #	391..1..	...3..1.	(16580) (.....)

pkt #	40							1		(16485) (.....)
pkt #	41				2			1.1.1	1.1.1	(16545) (.....)
pkt #	42		1		2			3	1	(16693) (.....)
pkt #	43				1		1		1	(16546) (.....)
pkt #	44	1.1			1	11	1	4.3	4.4	(16366) (.....)
pkt #	45		2	1						(16522) (.....)
pkt #	46									(16514) (.....)
pkt #	47		1							(16497) (.....)
pkt #	48									(16543) (.....)
pkt #	49				2			2	1	(16598) (.....)
pkt #	50	2	1	3	1		1	3	2	(16489) (.....)
pkt #	51			1	c			1.3		(16581) (.....)
pkt #	52					1				(16602) (.....)
pkt #	53									(16569) (.....)
pkt #	54			1			1.2			(16520) (.....)
pkt #	55			1				1.3	1	(16500) (.....)
pkt #	56									(16511) (.....)
pkt #	57			1						(16621) (.....)
pkt #	58		1			1	2			(16479) (.....)
pkt #	59	1.2		8	3.3	e.g.	3.3	3.1	1	(16391) (.....)
pkt #	60		1		a.c.	1		1		(16588) (.....)
pkt #	61			1						(16551) (.....)
pkt #	62									(16603) (.....)
pkt #	63	1								(16512) (.....)
pkt #	64					4	1	1.2	d	(16356) (.....)
pkt #	65		1					1.c	6	(16603) (.....)
pkt #	66					1		2		(16536) (.....)
pkt #	67			1					1	(16524) (.....)
pkt #	68	1								(16453) (.....)
pkt #	69							1		(16545) (.....)
pkt #	70		1	3			1			(16582) (.....)
pkt #	71									(16518) (.....)
pkt #	72			1	2		1	1		(16485) (.....)
pkt #	73	1		2						(16519) (.....)
pkt #	74		1			1				(16509) (.....)
pkt #	75								3	(16394) (.....)
pkt #	76	1	1						2	(16415) (.....)
pkt #	77	1		3		1				(16403) (.....)
pkt #	78	3								(16491) (.....)
pkt #	79			1		1		1		(16560) (.....)
pkt #	80	3	1							(16460) (.....)
pkt #	81	1	1			5			1	(16472) (.....)
pkt #	82	1					1			(16369) (.....)
pkt #	83			1			1			(16445) (.....)
pkt #	84					1			2	(16454) (.....)
pkt #	85				1					(16485) (.....)
pkt #	86					1		1		(16538) (.....)
pkt #	87			1	2			1.1		(16587) (.....)
pkt #	88	1		2	1.1		1			(16384) (.....)
pkt #	89	1		2				1	7.1	(16346) (.....)
pkt #	90	4	1		3		1	2	2	(16385) (.....)
pkt #	91			1	23	1		1		(16376) (.....)
pkt #	92	1	1							(16456) (.....)
pkt #	93	1		1	3	2	1	1	1	(16378) (.....)
pkt #	94								1	(16549) (.....)
pkt #	95	2	1				1			(16454) (.....)

pkt # 96211	(16394) (.....)
pkt # 97	2.....	1.....	1.....	1.....	(16397) (.....)
pkt # 981.....	.2.....	.1.....1.....	(16462) (.....)
pkt # 99	.1.....	1.f...1.	.3.1.3.1.....	(16446) (.....)
pkt # 100	.1.....	.1.....1.....	(16429) (.....)
pkt # 1011.1..2.....	(16501) (.....)
pkt # 102	(16507) (.....)
pkt # 1031.1.1.1.1.	(16435) (.....)
pkt # 1042111	(16418) (.....)
pkt # 105	1.....	1.....	2.....	1.....	(16474) (.....)
pkt # 106	.1.....1.....	(16475) (.....)
pkt # 107	.2.....	.3.....	.13.....1.....1.....	(16404) (.....)
pkt # 108	.1.....1.....	(16392) (.....)
pkt # 109	.1.....1.....	.1.....	(16343) (.....)
pkt # 1103..3..	(16429) (.....)
pkt # 1111.1.3.1.	(16540) (.....)
pkt # 1121	(16473) (.....)
pkt # 113	3.....1.	1.....	1.....	1.....	3.....	(16460) (.....)
pkt # 114	.1.....	.1.....2.....	(16437) (.....)
pkt # 1151.....	.1.....	.11.....	1.1.....2	.61.....	(16410) (.....)
pkt # 116	.1.....	2.....	.1.....1.....	11.1.....	(16383) (.....)
pkt # 1173.....	.11.2..1.....1.....	(16490) (.....)
pkt # 1184..2..12.4..1.3..1..	(16469) (.....)
pkt # 1191.1.1.1.1.1.1.1.	461b51..	(16302) (.....)
pkt # 120	87479c684.41.2	1..19.g1	62.11113	(16371) (.....)
pkt # 121	1.....	6.....1.	1..1..	2.....	1.....	4.....	1..1.....	(16439) (.....)
pkt # 122	.1.....11	.1.....	.1.....	.2..12.2..1..	16...24	(16339) (.....)
pkt # 123	.1.....	2.3.....	3.5.....	1.11....	1.21.1.	(16353) (.....)
pkt # 1241.....	.5.....6.5....6.31...	.1.1...	(16397) (.....)
pkt # 125	.111...	.1..2...	11333...	.1..1..	1..11...	.1.....1..	(16340) (.....)
pkt # 1261.6.....3..	.12.2..4..	.1..1..	.1.11..	(16446) (.....)
pkt # 1272.2.	.13.1.1.4.1.2.1.4.	(16371) (.....)
pkt # 1281.22311.3	(16399) (.....)
pkt # 129	1.....	2.....	2.....	1.....	(16453) (.....)
pkt # 130	.2.....2.....	.1.....	.1.....	.2.....	.2.....	(16321) (.....)
pkt # 131	.2..2.2	.2151.11	1.73....3.....1	.2.....	.2.....	(16330) (.....)
pkt # 132	.3.32...	.1.....	.1.5....	21.2....	.1.4....	.4.....1.2....	(16227) (.....)
pkt # 133	.1.....	.2.....	.2.....	.1.....	.2.....	.3..3...	.1..1...	(16367) (.....)
pkt # 1343..1..1..	(16479) (.....)
pkt # 1352.....1.1.11112.	(16392) (.....)
pkt # 1361111	(16435) (.....)
pkt # 137	2.....	3.....	7...2..	3...2..	(16495) (.....)
pkt # 138	.1.....	.1.....2.	.1.....	.2.....	.2.....	(16334) (.....)
pkt # 139	.4.....	.3.....	.3.....2.1...	.1.....	(16244) (.....)
pkt # 140	.1.....	.2.....	1..1...2.....	.1.....	(16336) (.....)
pkt # 141	.4.....12.....2.3...	.3.....	.2.....	.1.....	(16316) (.....)
pkt # 142	.3.....	.2..1..	.2.....3.2..	.1.9..	.2..3..	(16300) (.....)
pkt # 143	.1.1..2.....	.1.....	.2.....1.3.4.	(16399) (.....)
pkt # 1445	.3.3	.1.....23121.1	(16228) (.....)
pkt # 145	1...1..	.3..	2.....	1.....	1.....	2.....	(16442) (.....)
pkt # 1463.....	.6...2	.1.....	.3.....2.....	.1.....	(16377) (.....)
pkt # 147	1.....1.....2.....1.....	.1.....	(16365) (.....)
pkt # 148	.1.6....3.3....	.1.....	.1...2	.1.....	.2.....	(16228) (.....)
pkt # 1491.....	.2.....1.....	.1.1...	(16442) (.....)
pkt # 15011.1..	.2.....1..	.2.7..	(16380) (.....)
pkt # 1513..6..	.1..1..3.....	.3.....2:1.	(16464) (.....)

pkt # 1522..121255.51..2.41..1	(16288) (.....)
pkt # 153	6...32.	5...51.	1.....	2.....	1.....	(16222) (.....)
pkt # 154	.1.....	.1.....	.1.....1.....	.2.....4.....	(16435) (.....)
pkt # 1552	..3...8	..2....	..1....	..1...1	9.9...21	..1....	(16349) (.....)
pkt # 156	21.4...	21.....	.6.9...	13.5...	..13....1.5...	21.1...	(16184) (.....)
pkt # 1573...3...	.2..3...2...1...	(16284) (.....)
pkt # 1583..2..4..	..2.2..1..	..2..2..2..	..2..1..	(16330) (.....)
pkt # 1591.3.1..1.3.6.1.	(16349) (.....)
pkt # 160211.151.45...11	(16234) (.....)
pkt # 161	3.....	2...11.	2.....	2.....	1...2..	2...4..	9....2.	(16316) (.....)
pkt # 162	.3...2	.2.....	.2...1.	.2.....	.2.111.	.3.....1.....	(16215) (.....)
pkt # 1631.....	.1.....	.1.....	.1.....1...1	(16347) (.....)
pkt # 164	.e.f...	..2....	..2...2	..5....	..1....	..2....	..4....	..2....	(15866) (.....)
pkt # 1653...1...1...	32.1...	.1.2...	.1.....1...	(16176) (.....)
pkt # 1661..	..1..2..2..	..2.5..	..1.3..1.3..6..	(16392) (.....)
pkt # 167	..13.3.3.12	8.5..3.5	1.....4c	.41b6744	36.3..3.	(16109) (.....)
pkt # 1684516111	(16267) (.....)
pkt # 169	2.1...11..	1.....	1.....	(16346) (.....)
pkt # 170	.1.....1.....4.....1.	.1.....	(16345) (.....)
pkt # 171	1.6.....3	5.....1	1.2.....	.1.....	1.....	.1.....	.2.....	(16197) (.....)
pkt # 172	1.1...1	..1....	..3....2....	..2....	.1.2..1	(16207) (.....)
pkt # 173	.3..4..	.2..4..	.1..2..	..1...	..2...	..1...	..1...	..1...	(16152) (.....)
pkt # 174	..1.4..4..1..1..1..	..1....	(16291) (.....)
pkt # 1751.1.1.1.1.	(16292) (.....)
pkt # 1762113.64.a	(16408) (.....)
pkt # 177	2.....	1.....	2...1.	2...12.	(16293) (.....)
pkt # 178	.2.....	.1.....	.3...11.....	.3.....2.	.2...1.	(16380) (.....)
pkt # 1791...11.....	.1.....	.1.....	.1.....	1.2...1	(16337) (.....)
pkt # 180	..4....	1..1...	..4....	3.....	1..1...3....	.4.4....	(16223) (.....)
pkt # 181	..1...1...1...3...	..1...	(16292) (.....)
pkt # 1821..12..2..1..	..2.1..	..1..8..	(16337) (.....)
pkt # 183	..2..1.	..1..2.1.1.2.	(16276) (.....)
pkt # 184231223	(16281) (.....)
pkt # 185	5...1.	5...7..	5.....	3...4..	4.....	2.....	(16150) (.....)
pkt # 186	.5.....	.4...12...1	.1...1	.5...1.1.....	(16292) (.....)
pkt # 187	..4...12....	.1.....	.5.....	.1.....	.1.....	..2....	(16283) (.....)
pkt # 188	..4....	..1....	.1.3...	13.9...	.1.6...	.1.5...	.5.7...	..4....	(16218) (.....)
pkt # 189	.2..6...	.1..3...	.21.3...6...	.11.6...	.11.f...	..2.a...	.1..2...	(16140) (.....)
pkt # 190	..1..1..	..3..1..4..	..1.4..	..2..4..	..1..5..3..1..	(16303) (.....)
pkt # 191	..1..1.1....2.2.2.1.1.2.	(16322) (.....)
pkt # 19213.631.526.31	2...8.6	.1.1.c.b	(16296) (.....)
pkt # 1931	4.....	3...7..	3...81.	(16401) (.....)
pkt # 194	.5.....1	.1.....2.....	.2.....	.2...1	(16139) (.....)
pkt # 195	.1.....	.1.....1.....1.....	(16321) (.....)
pkt # 196	.1..1...	.1.1...	..4....	.3.8....	..2....	..1....	.1.5...	(16250) (.....)
pkt # 1973...	.1..4...	.11.3...5...2...	.1..2...2...	(16225) (.....)
pkt # 198	..1.1..2..	..6.7..4..	..2.6..	..8..5..3..	..41.2..	(16275) (.....)
pkt # 1991.1.2.1.3.6.2.1.	..6..4.22.3.	(16270) (.....)
pkt # 2001..11..1	1.....	8...123	2...7.d19.945.b	(16278) (.....)
pkt # 201	1...1..	1.....	2.....1	.2...1	1a.....	(16338) (.....)
pkt # 202	..3.....	..3.....	1.3.....	..2....2...1	.6...3	(16014) (.....)
pkt # 203	6.8.....	..1.....13....	..2....	..2....	1.2....	514...5	(16065) (.....)
pkt # 204	.3.6....	..2....	.2.4....	.2.5....	.1.3....11....	..1....	(16080) (.....)
pkt # 2052...	..3...	(16303) (.....)
pkt # 2062.3..	..3.2..	..2..2..2..1..	..1..2..1..	(16416) (.....)
pkt # 2072.1...4..	..11.1.2.5.42.a.3..	(16212) (.....)

pkt #	208	.96.1j.@	b4.e1dg731@23.a24...2.4	(15298) (.....)
pkt #	209	21.....	6.....5.	3.....	8....16.	7.....6.	3....1..	2....111	(16287) (.....)
pkt #	210	.2.....	.1.....	.3.....	.1.....	.1.....	.2.....	.1.....	.3....1.	(16186) (.....)
pkt #	211	..1....	..3....1	1.2....1	.14....	2.a....1.....	(16266) (.....)
pkt #	212	@1.3....	b..5....	..11....	...2....	13.5....	21.4....	3..2....	...1....	(15927) (.....)
pkt #	213	.3d.@...	.12.9...	122.4211	...2...1...	.1.....	(15771) (.....)
pkt #	2141..	...1.6..	..13.3..	..1..4..1..11.8..4..	(16305) (.....)
pkt #	2152.4.	...51.7.	...21...1.2.2.1.	(16116) (.....)
pkt #	2161..131111.33.1	(16190) (.....)
pkt #	217	1.....	5....21.	4....32.	2....1.11	2.....	6....2.	(16389) (.....)
pkt #	218	15.....2	.4....1.	.7..2.@3	.4..1.8.	.112.1..	.131.2..	.5.....1	.1...2.	(16170) (.....)
pkt #	219	@.i.....	4.a.....	.12....3	..2....	2.6.....	..31...1	1.4....	1.1....	(15742) (.....)
pkt #	220	3.13:...	...2....	.3.7....	...4....	.1.1....	13.6...	11.7...	1.....	(16094) (.....)
pkt #	221	.15.a...	.c..6...	.6.....2...	..2.7...	..1.8...	.1..2...	(16005) (.....)
pkt #	22212.	...f:h1.	..2b.e..	..11.4..1..1..	...2.4..	..16.d..	(16135) (.....)
pkt #	223	...9..2.1.1.	...1.2.	...4.7.	...2a.c.	...43.7.	(16112) (.....)
pkt #	2241	1.....1.41.212.2241	(16293) (.....)
pkt #	225	3.....	1.....	2.....	1....1.	4....3.	1.....	7....1..	(16267) (.....)
pkt #	226	.2.....1.	.4.....	%%%%%	&^&^^&&&	&%&&&^&&	&&^^&&&^	&&&&&&\$&	(16276) (.....)

system main menu:

- a = analysis experiment data
- b = browse through experiment for hard errors
- h = histogram of experiment channel
- n = next experiment file
- q = quit

option [a]?:

ARQ experimentfile [/users/rose/data/realexpt.001]: system main menu:

- a = analysis experiment data
- b = browse through experiment for hard errors
- h = histogram of experiment channel
- n = next experiment file
- q = quit

option [a]?:

channels are: cc23 cc133 info bc16 cc35 cc171 bc8 info

i am looking at experiment file: /users/rose/data/realexpt.11

pkt #	021.2	1.....1.	1..1.g.06a.a	.af.03.5	9*0\$99b7	8888^888	88^^8*^	(15894) (.....)
pkt #	1	6.7e.251	9..1.a61	..0.1...	c....4a5	j....00.	j....9h.	8....b63	52.....2	(15695) (.....)
pkt #	2	3b....88	3b....83	.41..1.2	.5....21	14....21	.3.....	.5....14	11....11	(15797) (.....)
pkt #	3	615....b	633....6	348....4	..2.....	a.c.....	g.d....4	1.3....1id..	(15551) (.....)
pkt #	4	9e.f7...	80501..1	27.7....	166d....	361d....	3213....	cb2b....	.4.7....	(15386) (.....)
pkt #	5	.d114..2	.1.1.2..	.4e1j...4...	.1123...4...	5:4.6.5.	.491i...	(15731) (.....)
pkt #	6	..3615..	..14.5..	..7528..2..	..38.8..	..33.7..	...617..	..1111..	(15824) (.....)
pkt #	7	...17.d.	...46.e.	...92.7.	...91.7.	1..31.6.	c652i2e1	...497e	%^^%^^%	(15725) (.....)
pkt #	82.818.7	%%%^^%	888888^	8^8^8888	8%88888*	88^^*88	888888\$8	(15841) (.....)
pkt #	9	3....11.	4....11	3....622	.2....1.	3.....	1.....	6....22	@...4c1	(16027) (.....)
pkt #	10	38...25	15...54	.4.....	33...34	12...2.	13.....	.f....4c	2f....d7	(15854) (.....)
pkt #	11	1.1.....	948...2	d.a.....	j.h....i	2.3....4	1.6....4	414...2	d1i.....	(16034) (.....)
pkt #	12	f114....	b417....	21.2....	..13....	2124....	.1.1....	5e3j.2.	a710....	(15849) (.....)
pkt #	13	..1.3...	.1519...	.15.5...	..434...	.2716...	.5alb...	.8915...4...	(15919) (.....)
pkt #	1411312.1	..20204.	..e8.g2.	..08.c..	..f8.8..	..3.22..	(16097) (.....)
pkt #	15	...4324.	...15.5.	48.5..3.	232.1i.2	...j0200	...44.9.	...j51a.	...0..6.	(15952) (.....)
pkt #	16	.45...12	.cgl..1.	.\$ei....	.2.120.0	.50.54.6	.g525...	.42\$91..	1..41j38	(15838) (.....)
pkt #	17	c.....9.	a.....7.	8....74.	4....42.	5....752	8....511	5.80.6..	2.\$5.7.1	(15732) (.....)
pkt #	18	3i....7a	1a....64	1a....7b	1b....a6	.1...1.	.b....6	.5....32	18....14	(15239) (.....)
pkt #	19	8.d...2	3.3...8	128...b	1434...4	368f1b12	735....	..3.....	a1g.....	(15780) (.....)
pkt #	20	12.9....	5b36....	2..4....	1115....	51.d....	3121....	.424....	.4.4....	(15901) (.....)
pkt #	21	.5..2...2..	.44.8...	.41.7...	...2...	.12.5...	..1.1...	..4.5...	(15825) (.....)
pkt #	22	..5..3..	..7a2d1.	..1335..	..32.3..	..941b..	..2..3..	..22fa06.	..1344\$62	(15932) (.....)
pkt #	23	...9g7\$2	...d.161	...111.	...3f2i.	...64.b.	...21.2.	...5637.2.	(15339) (.....)
pkt #	24	1...23.2	1...212.	2....52	3...12.3	...241b5.413.647.d	(15800) (.....)
pkt #	25	3.....1.	5....111	8....26.	7....16.	b....382	4....1..	f.....1.	6....722	(16003) (.....)
pkt #	26	131...2	3c%...1	.hb...c7	38...1.2	.2...2.	7e1...9h	22...11	.b...3a	(15492) (.....)
pkt #	27	134.15.4	..5.....	2.5....2	2.3....3	..8....1	31d1...1	..4.....	325....a	(15769) (.....)
pkt #	28	2626....	1..3....	1625....	171f....	a31a....	ad90...5	364e....	9625...2	(15790) (.....)
pkt #	29	.b316....	31472.1.	.48.7...	..2126...	..218...	..1439...	..25.5...	..34.3...	(15747) (.....)
pkt #	30	@0e0\$1dj	888888^88	^88888^8	^*888*88	8^88^8^8	\$*88^88	*8^*88^	88^88^8	(13736) (.....)
pkt #	31	...eb.f.	...3..72	...fg3a.	..2j70fh	113ce.4.	1c25.31.	7..ce3d.	...224.	(15638) (.....)
pkt #	32	...1315212	\$.1...3.	5...1106	..131..3.	...23.9	...1.12	..1...a.e	(15811) (.....)
pkt #	33	5.....	^%%^^%	*8*88%8*	*8888888	^8^8888	8%8^8888	*8%^^*8^	888888%8	(15937) (.....)
pkt #	34	3d...179	.2...2.	.1....11	19....5	1h....bi	.6....a1	.5....1	14.71..1	(15703) (.....)
pkt #	35	5.4...5	216...4	316....	c.j...5	2.81...6	119...1	i1c...9	7.\$...\$	(15844) (.....)
pkt #	36	51.7...	3318....	2124...1	43.9....	2428....	152a....	.1.8....	32c7....	(15866) (.....)
pkt #	37	...12...	41423.6.	11518.g1	.77.f...	.ed3i...	..1.4...	.2339...	...12...	(15833) (.....)
pkt #	38	..2413..	..221b..	..544fa.	..905e9.	g.401052	..bh1j..	..88.9..	..7.11..	(15901) (.....)
pkt #	39	...3c.g.	...7i10.	...i214.	...c1512	1...0d\$	16.18004	...8549.	...5g1c1	(15712) (.....)

Table with columns for packet numbers (pkt #) and alphanumeric data strings. Each row represents a packet, with the first column being the packet ID and subsequent columns containing various alphanumeric sequences. The table ends with a list of corresponding numerical IDs in parentheses, such as (15780), (15592), etc.

ARQ experimentfile [/users/rose/data/realexpt.001]: system main menu:

- a = analysis experiment data
- b = browse through experiment for hard errors
- h = histogram of experiment channel
- n = next experiment file
- q = quit

option [a]?:

channels are: cc23 ccl33 info bc16 cc35 ccl171 bc8 info

i am looking at experiment file: /users/rose/data/realexpt.12

pkt #	0&1..*2..&1..	3...^2.4	..14&8..&4..	.121^4...	..1.&4..	(15329) (.....)
pkt #	1	.1...&4.^..	...1.&e.&1.	63.1.&4.1^2.	...1.&3.	27...&4.	(15207) (.....)
pkt #	2	...122&4	...224^1	.458..&5&3	.4...1&2	5...1.&3	...32&21^7	(15050) (.....)
pkt #	3	2.1....&8	9.....*	8....a.&8	6.....&8	6.1...1.&8	1....41&8	6.....9&8	2.....3&	(15118) (.....)
pkt #	4	^8...31	^1....1	&43...22	^21..532	&6.....2	^9...47	^7.....	&4.....	(15129) (.....)
pkt #	5	.&2....	.&2....1	.&3....	.&2....	.&4....	.&3....	.^*....	.&6....1	(151&2) (.....)
pkt #	6	.4&8...&4	6.*6...4	1.&6e...	1.&22...	...&.a...	.b&9.13.	e1&71...	12^a2...	(15039) (.....)
pkt #	7	.42&4.81	.4c^42..	..1^35..	.1.^3...	...&5...	.2.&8..a	.1.^3...	.1.*4...	(14856) (.....)
pkt #	8&4..*3..	..1.&....	...1^2..	...1&4..&21.	...4^3..&a..	(15267) (.....)
pkt #	9	2..7.&..	...8.^2.	...1.&1.&3.&8.	1....^..	2.2.6&4.	..12.&1.	(15142) (.....)
pkt #	10&34.^2&54&a	2...16&.5&g1&4^6	(15172) (.....)
pkt #	11	4...12&8	4.....*	c....11&8	4.....&	1.....&	3...2.&8	1...1.8	2...4.&	(15176) (.....)
pkt #	12	^4...26	^26...3.	&2.....	^3...2.	&3...1	^3...2.	^2.....	&4.....	(14634) (.....)
pkt #	13	3&3.2..6	3&4...ae	.&2...4	3&6....	9&3...3	3&h2...7	.^5...2	.&2....	(14950) (.....)
pkt #	14	4.&a...	2.*7...	..&....	..&5...	1.&3...	..&....	..&3...	1.*2...	(15094) (.....)
pkt #	15	...&4...	.1.^7...	..1^j...	...^1...	.1.&4...	...&1...	...^3...	...*1...	(15176) (.....)
pkt #	16	.1.2&5..	..1.*3..&1..	..1.^1..&14.	..a.&ad.	.d6b^a..	.46.&...	(15162) (.....)
pkt #	17	...2&8&9.	...11^5.1&5.&1.	...21&6i	..4.1^7.	eb...&..	51..2^41	(14878) (.....)
pkt #	18	.g.18..5	.1....1	j...2..1@4h9	.59a4b.2	e...g1eac	...1.d.3	(15766) (.....)
pkt #	19	j3..16&35	=...41..	1...11.	6b...13	3.36.12.	1.4.522	65...1.6	3..5.5..	(15409) (.....)
pkt #	20	751.11j1	14..1.3.	..2...3.	1...3.	.2..9.1.	.1...1.	532....	33.11...	(15847) (.....)
pkt #	21	..7...4.	1.4.13..	2.2&...	.8.3..1b	2.4.1cl.	7.9a...1	2&81...4	1.3.c74.	(15850) (.....)
pkt #	22	...1.132	...4.1..	145d...7	29.41h.2	52464...	gc.h..5.	34.342..	..14....	(16031) (.....)
pkt #	23	.j423...	.8..4...	.71....	.7..4...	8.3.2...	..1.2.7.	..1.1...	..2.3...	(15859) (.....)
pkt #	24	.1.112..6.4	..12192.	5.h2.b..	..7217.6	..f2.d1.	2179.b..	...1..	(16025) (.....)
pkt #	25	1..1..1.	...6.9.	...1e.e2	.15.4.5.	...55.3.	...43.1.	...24.b.	...21.2.	(16097) (.....)
pkt #	26	...1..11	.di...1.	j1.....	1...253	...5558	...5616	4...46.2	1...2h4h	(16154) (.....)
pkt #	27	31....1	4.n.2561	5.3....	@b...2.	3...7.f	...1a9d.	2.b&2...	112....	(16056) (.....)
pkt #	28	.2...3.8	15...5d2	.1..e.31	15.3..29	.14.2..3	6c...cg	.6...h8	.2b...12	(15928) (.....)
pkt #	29	..2....	.1....	1.5....	..12....	5.32...2	2.3...3	6.2...1	(16079) (.....)
pkt #	30	...3....	..16....	...2...	...2...	.2.4.2..7..1.	...1...	(16263) (.....)
pkt #	31	...7...	..1.2...	.32.4...	...6...	.11.3...	.11.1...	.31.3...	...2...	(16131) (.....)
pkt #	32b.12..	..c524..	..51.a...4...1...	..23.6..3...	(16188) (.....)
pkt #	33	...1.3.4.7.	...56.9.b73.9..	...151&44c	(16087) (.....)
pkt #	34e..15.31d..111.315..4b.8	(15955) (.....)
pkt #	35	8...2.	1.....	6...1.	9....3.	4.....	5.a....	1.1.3..	(16139) (.....)
pkt #	36	.3...12	.c...lc	.@...343...5.	.2...11	.7...2	(16210) (.....)
pkt #	37	1.3...2...1	5.9...	7.8...	7d61..1.	(16205) (.....)
pkt #	38	19.d...	.d.#2...	.1.5...	1@.i...	.8.5c...	...1...	8..3...	(15726) (.....)
pkt #	39	...2...	.5...	..7.h...	..3.b...1.2...	(16256) (.....)

pkt #	404..	...1.1..4..4..	..1.1...11.5..	(16287) (.....)
pkt #	412.1.5.1.2.3.3.	...81.7.	1111116112.	(16148) (.....)
pkt #	4232.152142.6	1...1.42311	(16055) (.....)
pkt #	43	8.....5.	2....11.	4...1.7.	7.....3.1..	3...12.	(16123) (.....)
pkt #	44	.5.....	.2.....	.1...3.8.....8	.i...16	.3.....	.1.....	(16161) (.....)
pkt #	45	2.5.....	1.3.....	1.2.....	.6...2	1.1.....	.1...1	(16023) (.....)
pkt #	46	11.4....1.2....	...4....	.729....	.2.c1..	...1....	.1.11..	(16061) (.....)
pkt #	47	..2.7...21..	...3...	..1.3...	..2.1...	...2...	...3...	(16020) (.....)
pkt #	48	..4..2..	..3..1..	...1.5..	.9..a17.13.	.1.1.3..	.2..1a4.	...1.5..	(16296) (.....)
pkt #	492.2.11.11.23312.1.21.4.2.2.	(16100) (.....)
pkt #	502	1....9.16.j	1...33151.41	(16147) (.....)
pkt #	51	4...2..	1...2..	3...1..	1...1..	3...1..	3...12.	3.....	9...12.	(16213) (.....)
pkt #	52	.5.....14....2	.3...1.	.2...1	.3.....	.9....4	.2.....	(16131) (.....)
pkt #	53	.156...4	1.7.....	.3.....	.2.4443	43b...28	4.5...2	.2.....	1.4....	(15851) (.....)
pkt #	54	11.5...	17.a...4.1	..24....	12.2...	...6....	1..2...	1214..12	(16067) (.....)
pkt #	551...	.21.4...	..7.i.15	.3a.fa..	...11...	...2...	...2...	(16099) (.....)
pkt #	5614.3..4..4..	...8.b..	14.3.4.3	202455..	..71556.	(15900) (.....)
pkt #	571.3.3.	30.....	11f.4.f.	..213.gj	...1..1.3.5.	(16157) (.....)
pkt #	58313	...17.a	.b4621.6	3.36.115	c...6615	3...2j.s	...j.s	(16192) (.....)
pkt #	59	2.....1..	3.....	8.....	1.....	4...1..	8.2...1.	5.213..	(16290) (.....)
pkt #	60	1.....1.	.11...1.	.6...5.	.4...5	.2...1	.4...2	.5...1	.3.....	(16105) (.....)
pkt #	61	..5...1	2.5..5.h	4.3.g3..	.1..5#0.	4c3...4f	#b51..33	020a...g	0.jj...d	(16166) (.....)
pkt #	62	.2102...	5d1fh...	.344d...	...80...	..1.c...	.1.3...	1..4...	42.5...	(15414) (.....)
pkt #	63	..1.2...	..3.4...	..1.4...	...3...	..1.4...1.3...	..1.2...	(16100) (.....)
pkt #	641..	..2.3..2..	...1.1..5..	322.15.2	\$544493b	aeb000f9	(16240) (.....)
pkt #	65	###\$###0	8^8^888	8^88^8^	8888^88	8^8*8^8	8^88*88	8^*888^*	*8^8^8^	(12111) (.....)
pkt #	664	1.....1.2.3	...1113	...1..11.5	...4314	(16133) (.....)
pkt #	67	2.....	.1...5.	a1...4b2	59...132	.b...2	3...2..	4.....	(16315) (.....)
pkt #	68	.4...1	.1.....	19.5..69	13.#943.	321eg00.	191.19ef	744..912	244...1.	(16101) (.....)
pkt #	69	..3.....	..1.....	..1.....	2.a...5	..2.....	..3.....	1.6....	2.3.8..1	(16069) (.....)
pkt #	70	042\$3..1	d53\$6...	.130\$...	1137h...	41.42...	1.....	.1.....	.1.4....	(14932) (.....)
pkt #	71	..2.2...2...2...	..1.7...4...	...2...	...2...	(16099) (.....)
pkt #	72	21.1.1.0	01..1..e	d1.211.8	292418.3	.063.3.b	.07.11.2	..i2.jj.	..i.4ae.	(15579) (.....)
pkt #	73	..13\$#98	..ge#05.	..5c\$#31	...\$#d04	...163.	..41.31.	1124.2..	1.111.3.	(14469) (.....)
pkt #	74	.1.3.4.3	.5.8.231	.s7d...	.0721.22	.7..21.6	.0.2.1.3	.011.112	..2.a..2	(15937) (.....)
pkt #	75	.38.7...	2.314.1.	..0j3el.	7.\$\$.0s.	9.\$#0b1	1130\$1.2	8.00016.	119h.1.	(15861) (.....)
pkt #	76	.6...34	.3.....	.3.....1	.1.....	.1.....	.2.....	.1.....	.4...3	(15927) (.....)
pkt #	77	2.1.....	1.6...1	.4.....	..1.....	1.3...1	..4....	..2.....	..1.....	(16163) (.....)
pkt #	78	5530\$...	0j0ji..1	4b71.0#	...1.#0.	2115....	...5....	111b....	12.5....	(15004) (.....)
pkt #	79	.11.3...	.22.4...	.1.6...	..76f9..	21.31.58	.1.3.12	...3...	...4...	(16095) (.....)
pkt #	80	...1.2..41.	..62.92.	51b5242.	5...4.b1..4..	(16145) (.....)
pkt #	812.5.2.5.1.132	..1.122223.	(16086) (.....)
pkt #	82211.41.8d.11	.191b113	..1..2.5	...41.47	(16200) (.....)
pkt #	83	2.....	8...31	n...70.	5...11.	1...11.	1...1..	2.....	1g...2.2	(16197) (.....)
pkt #	84	.21.....	.2.....	142...13	.7...43g	.2.5..1.	.3...1.	20d...7	981f3e0:	(16241) (.....)
pkt #	85	3.4.....	24b0..24	915.0#eb	2.1.59..	..3.....	1.5...1	..6c....	aa#4..71	(16088) (.....)
pkt #	86	.1.3...	.16ah...	a52610e0	.1.4....	22.4....	06h0e..4	5221.#j0	1626.2..	(16184) (.....)
pkt #	87	6c10bi.a	512.2.i#	.3..1...	..1.3...	..13ci..	3b#002c7	.1328.23	.1..1...	(14859) (.....)
pkt #	88	b83.11.22..1..	abqca481	..1.13.e	..3..5..52.	..58231.	(15689) (.....)
pkt #	89	...6.5.	...3.22	1.165.52	...1.5.	...1.4.	...1.493	3.81259.	...1.2.	(16061) (.....)
pkt #	90	.2.....22	3..96928	.7a32...1.2	3...1..1	b...1d20	(16083) (.....)
pkt #	91	1...1.1	7a...22.	a11a.f59	2....2.	2.....	93...131.	4.....	(16147) (.....)
pkt #	92	.4...2.	.3...21.	12.....5...2.	.9d....	0h4144b.	(16037) (.....)
pkt #	93	##\$eb#e	^%#%888^	8*888^8^	^8888^88	^^^888	88888^88	88^^88^	88^88888	(12828) (.....)
pkt #	94	13.8....	.3482...	41.5..4.	..111.2.	d7he0...	1.21.292	...3....	(16000) (.....)
pkt #	95	.11.2...	.63.61..	22j5c..8	...3...	..1.2...	384cdj..	22425.d1	.11.6...	(16166) (.....)

Table with columns: pkt #, 10 columns of alphanumeric characters, and 10 columns of alphanumeric characters. Rows range from 152 to 207. Includes asterisks next to some row numbers.

pkt #	264	60000047	b0000000	d0000000	10000006h	500000010	\$\$\$%^\$	^^&&&&^	&*&^&	(13535) (.....)
pkt #	265	fa000006	j7000004	i8000003	f5000006	01000003	%\$%^\$	&&&&&^	&^&^&&&	(13493) (.....)
pkt #	266	ab.00000j	2d100000	50j00000	5h800000	4j300000	\$\$\$%^\$	^^&&&&^	&^&&&^&&&	(13683) (.....)
pkt #	267	.b500000	06i90000	\$4f60000	02g40000	02f30000	^^\$%^\$	^^&&&&^	&&&&^&&*	(13640) (.....)
pkt #	268	03006000	02f90000	\$8j50000	05070000	03j40000	^^\$%^\$	^^&&&&^	^&&&&^&&*	(13573) (.....)
pkt #	269	00602000	00600000	\$40b0000	\$gfc0000	04d40000	%\$%^\$	&^&&&&^	&&*&^&&&	(13466) (.....)
pkt #	270	00409000	000ed000	00000000	&&^&&&^&	&&^&&&^	&^&&&&^	&^&&&&^	&&*(^&&&	(13459) (.....)
pkt #	271	0000694	00006j5	000003b5	00000a83	00000cda	%\$%^\$	^^&&&&^	&&&*&^&&&	(13396) (.....)
pkt #	272	2000000.i	70000003f	500000020	40000002b	50000002i	\$\$\$%^\$	^^&&&&^	&&&*&^&&&	(13553) (.....)
pkt #	273	06g00000	51000000f	070000006	jb0000002	0b0000002	\$\$\$%^\$	^^&&&&^	&&&*&^&&&	(13606) (.....)
pkt #	274	30d00000	2f000000%	5e100000j	40300000	30600000	\$\$\$%^\$	^^&&&&^	*&&&&^&&&	(13241) (.....)
pkt #	275	02b40000	00900000	\$5000000	05qc0000	02i.0000	%\$%^\$	&^&&&&^	&^*&^&&&	(13665) (.....)
pkt #	276	00704000	002g7000	\$06f5000	002i4000	002f.0000	^^\$%^\$	^^&&&&^	&&&&&&&&*	(13416) (.....)
pkt #	277	0004j400	\$0570000	\$00b0000	00040100	00030000	^^\$%^\$	^^&&&&^	&&*&^&&&	(13622) (.....)
pkt #	278	000f0d00	00009005	\$00.g100	00005009	0000ah80	^^\$%^\$	&&^&&&^&&	&&&*&^&&*	(13108) (.....)
pkt #	279	000003e0	00000609	000004e7	000004e2	000002j6	^^\$%^\$	&^&&&^	&&&*&^&&*	(13469) (.....)
pkt #	280	20000003b	d0000009g	60000000	900000050	20000008a	\$\$\$%^\$	^^&&&&^	&&&*&^&&&	(13329) (.....)
pkt #	281	00b000005	hf0000004	jd000000a	7.000000e	i2000000	\$\$\$%^\$	^^&&&&^	&&&&&^&&&	(13445) (.....)
pkt #	282	40b000000	0afd0000	djd00000	7g200000	1h.000000	\$\$\$%^\$	^^&&&&^	&^&&&^&&*	(13235) (.....)
pkt #	283	.6100000	05f90000	\$9h60000	05h80000	02e60000	^^\$%^\$	^^&&&&^	&^&&&^&&*	(13600) (.....)
pkt #	284	003f2000	002e5000	\$7h80000	005j6000	001j3000	^^\$%^\$	^^&&&&^	&&&&^&&&*	(13514) (.....)
pkt #	285	0004f300	0005i200	\$0050200	0006j400	00006630	^^\$%^\$	^^&&&&^	&&&*&^&&*	(13444) (.....)
pkt #	286	00007h60	00006j30	\$004e400	00003g90	00000d0c	^^\$%^\$	^^&&&&^	^&&&&^&&*	(13450) (.....)
* pkt #	287	.21.11..	.7..a...	.13.a...2...7...	\$\$\$%^\$	&&&&&^	&&&&&^&&*	(16051) (.....)
pkt #	2883.1.	411.a.10	3.....21..	\$\$\$%^\$	^^&&&&^	&^&&&^&&&	(15629) (.....)

system main menu:

- a = analysis experiment data
- b = browse through experiment for hard errors
- h = histogram of experiment channel
- n = next experiment file
- q = quit

option [a]?: