

Spring 2006

**Problem Set 1**

**Reliability and Availability Models, Computational Capacity, Linear Codes,  
Algorithm-Based Fault Tolerance**

**Issued:** Tuesday, February 7th.

**Due:** Tuesday, February 28th.

---

**Readings:**

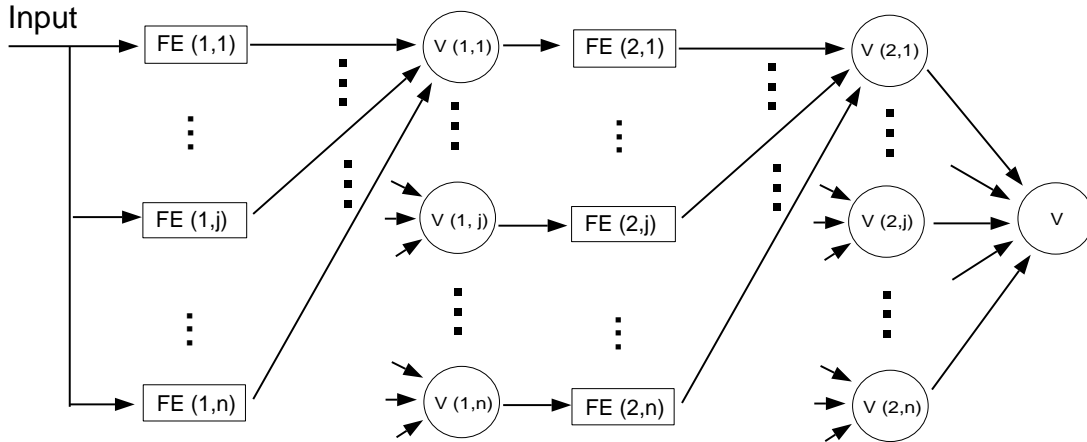
- (i) Johnson, Chapters 1, 2 and 3;
  - (ii) Siewiorek, Chapters 1 and 2;
  - (iii) Hadjicostis, Chapters 1, 2 and 3;
  - (iv) Blahut, Chapters 1, 2 and 3;
  - (v) Wicker, Chapters 1, 2, 3 and 4.
- 

**Problem 1.1**

Find the steady-state availability of a stand-by scheme that consists of three system replicas that fail independently. Assume that the rate of failure of a (functional) system replica is  $\lambda$  failures/second and that the rate of repair is  $\mu$  repairs/second (regardless of whether one, two or three system replicas are in a fail state). The overall stand-by system is considered available as long as at least one system replica is functional.

**Problem 1.2**

Consider the interconnection of voters and replicas of two functional elements,  $FE_1$  and  $FE_2$ , shown below. Voters are drawn as circles and denoted by  $V(i, j)$  ( $i = 1, 2, j = 1, 2, \dots, n$ ), and replicas of functional elements are drawn as rectangles and denoted by  $FE(i, j)$  ( $i = 1, 2, j = 1, 2, \dots, n$ ) where  $n$  is the number of replicas for  $FE_i$ . Assume that the reliability of each voter is  $R_V(t)$ , that the reliability of each functional element is  $R_{FE}(t)$ , and that failures in different voters/systems are statistically independent. Each voter  $V(i, j)$  receives inputs from the  $n$  preceding replicas of functional element  $FE_i$  and provides an input to exactly one replica of functional element  $FE_{i+1}$  in the next stage (if there is one). The last voter  $V$  receives inputs from all  $n$  voters preceding it and provides the final output of the system.



- Calculate (a bound on) the reliability of the final output of the interconnection above.
- Suppose that you only have access to three-input voters whose reliability is  $R_3(t)$ . How would you interconnect functional elements and voters so as to obtain a reliable implementation of the unit that consists of  $FE_1$  followed by  $FE_2$ ? What would be the associated (bound on the) reliability of the output?
- Can you generalize the analysis in parts (a) and (b) to an interconnection that involves  $k$  stages of functional elements and voters (instead of just two)?

### Problem 1.3

Von Neumann considered unreliable three-input voters that fail exactly with probability  $\epsilon$  and showed that as long as  $\epsilon < \frac{1}{6}$  we can reliably perform the voting function. More specifically, this can be done by having a depth- $d$  ternary tree of unreliable voters. In this tree, each of the  $3^d$  leaf nodes receives a triplet of inputs that are independently generated versions of the three inputs that we need to reliably vote upon. If each of these inputs is erroneous with probability  $p < \frac{1}{2}$  (independently, of course, from all other inputs since components are assumed to fail independently), Von Neumann shows that the output of the ternary tree of unreliable voters is erroneous with probability

$$p_{out} = f_{\epsilon}(f_{\epsilon}(f_{\epsilon}(\dots f_{\epsilon}(p)))) ,$$

where the iteration is taken  $d$  times and  $f_{\epsilon}(n) = \epsilon + (1 - 2\epsilon)(3n^2 - 2n^3)$ . Furthermore, for large enough  $d$  we can get  $p_{out}$  arbitrary close to

$$n_0 \equiv \frac{1}{2} \left( 1 - \sqrt{\frac{1 - 6\epsilon}{1 - 2\epsilon}} \right) .$$

Consider an extension of this setup to the case where one is dealing with  $u$ -input unreliable voters (where  $u$  is an odd integer) that fail exactly with probability  $\epsilon$ . Use a  $u$ -ary tree (where each voter is connected to the outputs of  $u$  other voters and where the leaf voters get as inputs the  $u$ -inputs that we need to reliably vote upon) and assume that each input to the leaf voters is erroneous with probability  $p < \frac{1}{2}$ , independently from all other inputs. By finding and analyzing the corresponding  $f_\epsilon(p)$ , find the maximum value of  $\epsilon$  and the corresponding  $n_0$ .

**Problem 1.4**

An  $(n, k)$  Hamming code over  $GF(2)$  is *perfect* if it satisfies

$$n = 2^p - 1,$$

where  $n$  is the total number of bits and  $p = n - k$  is the number of additional bits (the  $(7, 4)$  Hamming code is an example of a perfect code).

- (a) Find the parameters of a perfect Hamming code with codewords of size  $n > 7$ . What is the corresponding number of information bits?
- (b) Provide the parity check matrix for the code you found in part (a).
- (c) Provide the generator matrix for the code you found in part (a).

**Problem 1.5**

Find the length  $n$ , the dimension  $k$  and the minimum distance  $d$  for the linear code defined by the parity-check matrix below.

$$\mathbf{H} = \begin{bmatrix} 1 & 0 & 0 & 0 & 1 & 0 & 1 \\ 0 & 1 & 0 & 0 & 1 & 1 & 1 \\ 0 & 0 & 1 & 0 & 1 & 1 & 1 \\ 0 & 0 & 0 & 1 & 0 & 1 & 1 \end{bmatrix}$$

**Problem 1.6**

Recall that in the setup considered by Elias an unreliable two-input gate can be described probabilistically by a table like the one shown below.

$x$	$y$	out
0	0	$p_{00}$
0	1	$p_{01}$
1	0	$p_{10}$
1	1	$p_{11}$

The constant  $p_{ij}$  captures the probability with which the gate outputs 1 given that its inputs are  $x = i, y = j$  ( $i, j \in \{0, 1\}$ ). Assume that we are given an unreliable AND gate which satisfies  $p_{11} > 1/2$  and  $p_{00} = p_{01} = p_{10} = 1 - p_{11} < 1/2$  and consider the following scheme which (as discussed by Elias) cheats by allowing the decoding mapping at the decoder to be many-to-one under fault-free conditions.

Elias suggests that we can “encode” the  $k$ -bit input string  $\mathbf{X} = x_1x_2\dots x_k$  into the  $2k$ -bit string  $\mathbf{X}1^k$  (i.e.,  $\mathbf{X}$  followed by  $k$  ones) and the  $k$ -bit input string  $\mathbf{Y} = y_1y_2\dots y_k$  into the  $2k$ -bit string  $1^k\mathbf{Y}$  (i.e.,  $k$  ones followed by  $\mathbf{Y}$ ). After the unreliable gate ANDs (bitwise) the two encoded  $2k$ -bit strings, the decoder obtains a possibly corrupted version of the  $2k$ -bit string  $XY$  which can then “decode” into the desirable  $k$ -bit string  $(\mathbf{X} \text{ AND } \mathbf{Y})$ .

Ignoring the fact that the decoder is doing the actual computation, we still need to ensure that the decoder obtains a correct version of the  $2k$ -bit string  $XY$  (i.e., we need to additionally encode our inputs). Suggest a scheme to overcome this limitation. What would be the computational capacity of the unreliable AND gate for your scheme?

### Problem 1.7

Recall that in the context of algorithm-based fault tolerance (ABFT), Abraham et al. protect the corruption of a single entry of an  $n$ -dimensional vector  $\mathbf{v}$  (which could be, for example, the result of the unreliable matrix-vector multiplication of an  $n \times n$  matrix with an  $n$ -dimensional vector on a 1-D systolic array) by calculating two additional entries, essentially encoding vector  $\mathbf{v}$  into the vector  $\mathbf{c}$  given by

$$\mathbf{c} = \underbrace{\begin{bmatrix} 1 & 0 & 0 & \dots & 0 \\ 0 & 1 & 0 & \dots & 0 \\ \vdots & & & \ddots & \vdots \\ 0 & 0 & 0 & \dots & 1 \\ \hline 1 & 1 & 1 & \dots & 1 \\ 1 & 2 & 2^2 & \dots & 2^{n-1} \end{bmatrix}}_{\mathbf{G} = \begin{bmatrix} \mathbf{I}_n \\ \mathbf{P} \end{bmatrix}} \mathbf{v}$$

If the  $i$ th entry of the encoded vector  $\mathbf{c}$  is corrupted by an additive value  $v$ , then the parity check will result in the syndrome

$$\mathbf{H}(\mathbf{c} + v\mathbf{u}_i) = v\mathbf{H}\mathbf{u}_i = v\mathbf{H}(:, i),$$

where  $\mathbf{u}_i$  is an  $(n+2)$ -dimensional unit vector with a single “1” at its  $i$ th entry and  $\mathbf{H}$  is defined as

$$\mathbf{H} = \begin{bmatrix} \mathbf{P} & -\mathbf{I}_2 \end{bmatrix}.$$

(Note that  $\mathbf{H}(:, i)$  denotes the  $i$ th column of matrix  $\mathbf{H}$ .)

A student who took 586CH two years ago, suggests that single-error correcting capability can also be achieved by encoding using the matrix

$$\mathbf{G}' = \begin{bmatrix} 1 & 0 & 0 & \dots & 0 \\ 0 & 1 & 0 & \dots & 0 \\ \vdots & & & \ddots & \vdots \\ 0 & 0 & 0 & \dots & 1 \\ 1 & 1 & 1 & \dots & 1 \\ 1 & 2 & 3 & \dots & n \end{bmatrix}.$$

Do you agree with this statement? Explain. Are there any advantages/disadvantages in using  $\mathbf{G}'$  over  $\mathbf{G}$ .