Dependable Computer Systems and Networks

HW #2

R10943170 連德宇 **Due Oct. 3, 2022**

1. (10 points) Some systems are designed for reliability whereas others are designed for availability. Explain the difference between *reliability* and *availability*, and give an example of an application requiring high availability and one requiring high reliability.

Sol:

根據講義 p75&147&150&155

Availability: the probability that a system is functioning correctly at the instant of time t.

Reliability: the probability that a system will perform its functions correctly

application requiring high availability: Banking Systems

application requiring high reliability: Boeing 777

2. (20 points) For the following systems (A and B), identify which attribute (reliability, availability etc.) is considered least important. Justify your answers.
   A. An aircraft system has three computers voting on the results of every operation performed by the auto-pilot. If the auto-pilot fails, a warning alarm goes off in the cockpit to alert the pilot, who can then take over the manual controls of the aircraft and guide it to safety. However, the pilot does not interfere as long as the autopilot does not raise the alarm.
   B. An online trading website allows its customers to place bids on various items, and to track their bidding online. While it is acceptable for a user to not be able to place bids if the traffic is too high, it is not acceptable for a user who has placed bids to not track their bid's status and modify the bid. Also, as far as possible, the website should not display an incorrect value of the item's current bid, as this can cause users to over/under-bid for it.

   In each of the following descriptions (C and D), identify the fault, error and failure.
   C. A program contains a rare race condition that is only triggered when the OS schedules threads in a certain order. Once triggered however, the race condition corrupts a value in the program, which in turn is used to make a branching decision. If the branching decision is incorrect, the program will go into an infinite loop and hang, thus failing to produce any output.
   D. A radar system uses an array of processors to track its target in real-time. A soft error in a processor can lead to the processor computing an incorrect value for the target's location. However, the system can compensate for this effect by redundantly allocating the tasks to processors and comparing the results. But this compensation entails a performance overhead, which in some cases, can cause the system to miss the tasks' deadlines and lose the target.

Sol:

A:
我認為在此系統中，**reliability相較起來是比較不被重視的**，而availability則相對重要，因為在此系統中，駕駛員可以在決策系統故障時手動干預，若是availability出現問題，則會產生系統無法被操控的問題

B:
在這個系統中，**availability相較起來是比較不被重視的**，因為系統可能因為瞬時流量過高，導致無法使用，但是網站應盡可能不顯示項目當前出價的錯誤值，因為這可能導致用戶對其出價過高/過低，所以而reliability則相對重要。

C:

Fault:	branching decision is incorrect
Error:	go into an infinite loop and hang
Failure:	failing to produce any output

D:

Fault:	soft error in a processor
Error:	the processor computing an incorrect value let system have to compensation
Failure:	miss the tasks' deadlines and lose the target

3. (10 points) A telephone system has less than 3 min per year downtime. What is its steady-state availability?

Sol: $A_{ss} = \frac{365*24*60-3}{365*24*60} = 0.99999429223$

4. (20 points) A copy machines manufacturer estimates that the reliability of the machines he produces is 73% during the first 3 years of operation.

   (a) How many copy machines will need a repair during the first year of operation?
   (b) What is the MTTF of the copy machines?
   (c) The manufactures guarantee MTTR = 2 days. What is the MTBF of the copy machines?
   (d) Suppose that two copy machines work in parallel and the failures are independent. What is the probability of failure during the first year of operation?

Sol:

(a) $R(t) = e^{-3\lambda} = 0.73$, $\frac{\ln(0.73)}{-3} = 0.10490358161$
故約 10.49035%的印表機會在第一年後需要維修
(b) $MTTF = \frac{1}{\lambda} = 9.53256299377$years
(c) $MTBF = MTTF + MTTR = 9.53256299377 + \frac{2}{365} = 9.53804244582$years
(d) $R(t)^2 = (0.10490358161)^2 = 0.01100476143$

5. (10 points) Devise an original example (different from the lecture examples) to illustrate the difference between faults, errors, and failures. As you illustrate these concepts, relate them to the three-universe model.

Sol:

Fault:	電力系統出現人為失誤(physical universe)
Error:	導致無法上網(informational universe)
Failure:	交通控制系統錯誤，導致發生車禍(in the external universe)

6. (10 points) Why redundancy techniques used in hardware system cannot be used for software fault tolerance. If you are employed as a software quality engineer, what techniques you will prefer?

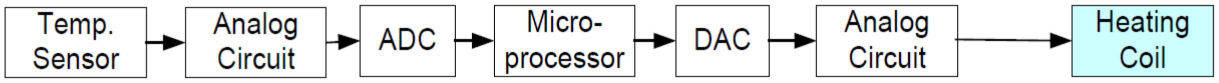
Sol:因為硬體可以將電路拆分為不同的 module 去做時序檢查分析，但是軟體上的問題通常息息相關，較難以分開去做檢查除錯。

而以我前陣子看到的論文舉例，他表示較前面的 ML layer 錯誤影響的 weight，相較之後出現的影響更大，因此我想若是使用 N-version programming 的概念，並搭配 XOR 做 voting，就可以有效避免這類型的 software fault。

7. (20 points) The company that you work for is designing an industrial controller that maintains the temperature of a fluid during a chemical reaction. The non-redundant controller (figure below) contains: (1) temperature sensor; (2) analog circuitry to process the temperature sensor's output signal; (3) analog-to-digital converter (ADC); (4) microprocessor (including hardware and software); (5) digital-to-analog converter (DAC); (6) analog circuitry to process the output of the DAC; and (7) heating coil to control the temperature. You have been asked to develop at least two approaches for making the controller tolerant of **any two faulty components**. The term "component" means one of the blocks of functionality listed above, **excluding the heating coil**.

   (a) Show block diagrams of your two approaches and compare them qualitatively. Note that your designs should be able to handle faults of any two components, including any two same components (e.g., 2 ADCs) and any two different components (e.g., 1 ADC and 1 temperature sensor).

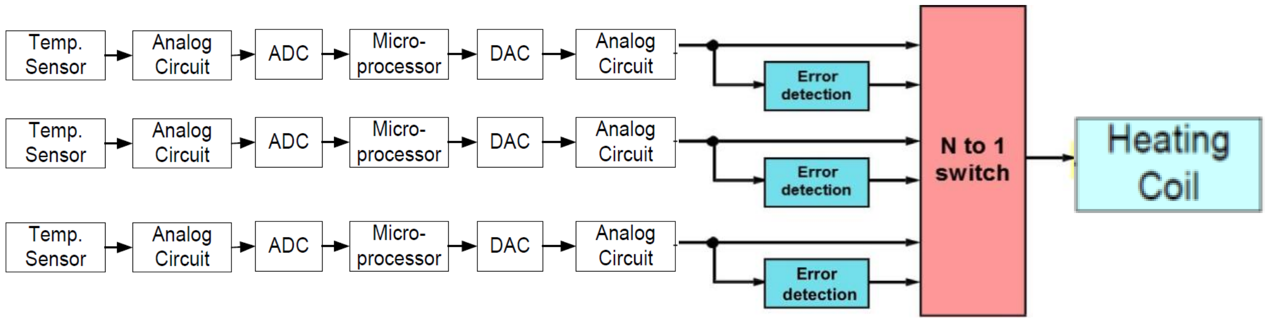
   (b) Which approach would you recommend for implementation and why?



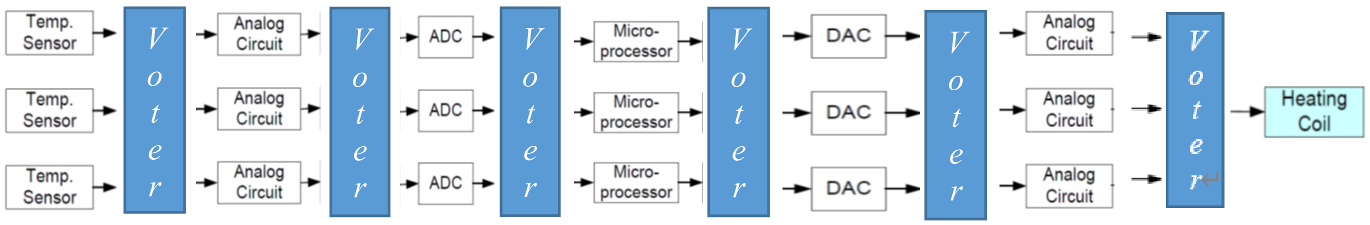
Sol:

(a):

1.



2.



Both methods are hardware redundancy, the first one is from p91, it can detect the error and remove the faulty module from operation, the second one can achieve fault tolerance without repair

(b):我會選擇使用第一種方法，因為既可以偵測錯誤，亦可以排除，且 PPA 理論上應該較第二種來得好

8. (20 points) Moon Systems, a manufacturer of scientific workstations, produces its Model 13 System at sites S1, S2, S3; 20% at S1, 35% at S2, and the remaining 45% at S3. The probability that a Model 13 System will be found defective upon receipt by a customer is 0.01if it is shipped from site S1, 0.06 if from S2, and 0.03 if from S3.

   a. What is the probability that a Model 13 System selected at random at a customer location will be found defective?

   b. Suppose a Model 13 System selected at random is found to be defective at a customer location. What is the probability that it was manufactured at site S3?

Sol:
(a):

$$0.01 * 0.2 + 0.35 * 0.06 + 0.45 * 0.03 = \mathbf{0.0365}$$

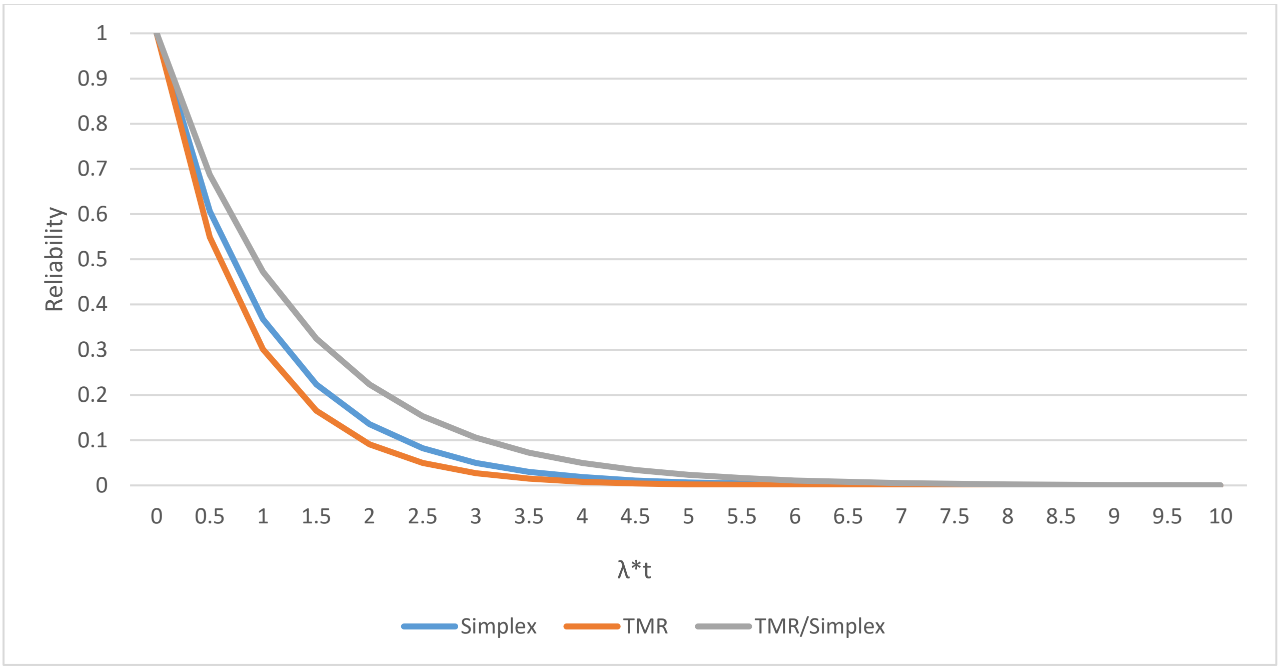
(b):

$$\frac{0.45 * 0.03}{0.01 * 0.2 + 0.35 * 0.06 + 0.45 * 0.03} = \mathbf{0.36986301369}$$

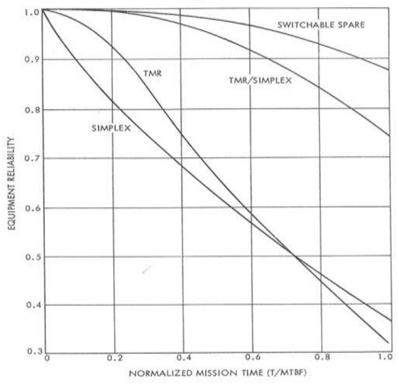
9. (10 points) Show that the reliability of TMR/Simplex is always better than either TMR or Simplex alone.

Simplex	- single unit
	- $MTTF = \frac{1}{\lambda}$
TMR	- three or N units with a voter
	- $MTTF = \frac{5}{6\lambda}$
TMR/Simplex	- after the first failure, a good unit is switched out with the failed unit
	- $MTTF = \frac{1}{3\lambda} + \frac{1}{\lambda} = \frac{4}{3\lambda}$

So the $MTTF_{\text{TMR/Simplex}} > MTTF_{\text{Simplex}} > MTTF_{\text{TMR}}$



Reliability of Redundant Systems



NASA Space Vehicle Design Criteria (Guidance and Control)

Spaceborne Digital Computer Systems

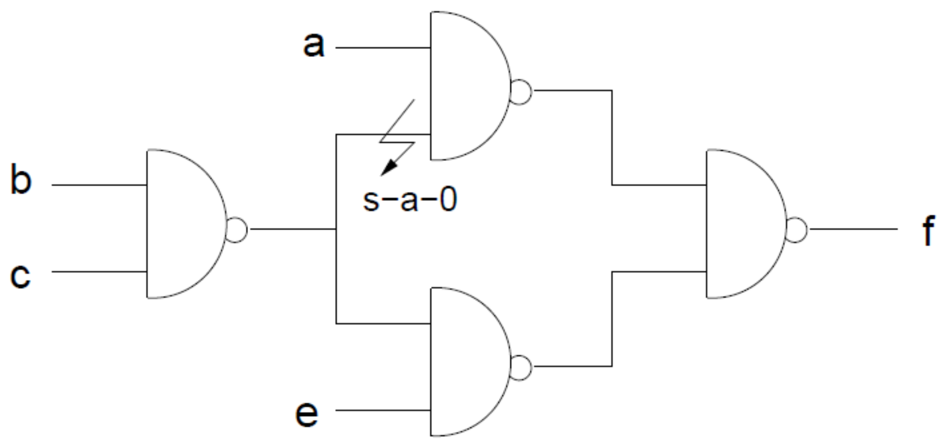
NASA SP-8070

W. Hoffman, Aerospace Systems , Inc.
A. Hopkins, Jr., MIT
J. Green, Jr., Intermetrics, Inc.

March, 1971

且根據 NASA space vehicle design criteria 這份文件中可知 the reliability of TMR/Simplex is always better than either TMR or Simplex alone

10. (10 points) Find **all** tests for the stuck-at-0 fault on the marked line



a	b	c	e	f	s-a-0
1	1	1	1	0	
1	1	1	0	0	
1	1	0	1	1	
1	1	0	0	1	0
1	0	1	1	1	
1	0	1	0	1	0
1	0	0	1	1	
1	0	0	0	1	0
0	1	1	1	0	
0	1	1	0	0	
0	1	0	1	1	
0	1	0	0	0	
0	0	1	1	1	
0	0	1	0	0	
0	0	0	1	1	
0	0	0	0	0	

(a, b, c, e) = {(1, 1, 0, 0), (1, 0, 1, 0), (1, 0, 0, 0)}