一个多节点声纳系统中同步时钟机制的可靠性评估和系统优化问题

冯绍庭 520021911362

摘要：本文以某种包含主从节点和时钟总线的多节点声纳系统作为研究对象，利用统计学和蒙特卡洛方法，探究该系统中同步时钟机制的可靠性、平均寿命以及相对应的最优节点数量。由于系统随机性较强，需要采用蒙特卡洛方法大量模拟。本文将系统分层级建模，采用变步长马尔科夫链，探究系统性质。分析系统失效原因后，本文提出双总线系统作为优化方案，并且对原系统给出可用性理论数值进行比较。

关键词：可靠性评估，系统平均寿命，蒙特卡洛算法，变步长马尔科夫链

Reliability evaluation and system optimization of synchronous clock mechanism in a multi-node sonar system

ABSTRACT: This paper takes a certain multi-node sonar system including master-slave nodes and clock bus as the research object, and uses statistics and Monte Carlo methods to explore the reliability, average life and corresponding optimal nodes of the synchronous clock mechanism in the system. quantity. Due to the strong randomness of the system, a large number of simulations using the Monte Carlo method are required. In this paper, the system is modeled hierarchically, and the variable-step Markov chain is used to explore the properties of the system. After analyzing the system failure reasons, this paper proposes a dual-bus system as an optimization scheme, and compares the availability theoretical values of the original system.

Key words: reliability evaluation, system's average life, Monte Carlo algorithm, variable-step Markov chain

目录

1 引言

本课题假设一个多节点声纳系统，每个节点都是分布式部署的，一共有 n 个节点。一旦电子元件出现故障，就无法进行人工维修。为了保证系统的正常运行，各个节点必须保持严格的时钟同步，并通过时钟总线传输系统时钟信号。只有一个节点可以工作在主时钟模式，其他节点必须工作在从时钟模式。当主模式节点的时钟电路发生故障时，可以自动退出主模式，其余节点按照一定的机制随机选择一个节点接管主模式下的工作。

因此，我们提出了如图 1 的冗余设计。在时钟正常同步的前提下，有 3 个节点正常工作才能使整个系统正常运行，节点总数 n 可以大于 3；当正常工作的节点数小于 3 时，系统失效。虽然这样尽可能地提高了电路的可靠性设计，但节点电路仍有可能出现阻塞时钟总线的故障，导致系统故障。因此，我们需要对系统进行定量评价，求解出最优的 n 值。

2 问题描述与模型抽象

2.1 整体思路

采用**变步长算法**，在每一个切换器故障的时刻，判断相应节点的状态，并根据一定条件得到整个系统的状态。求解系统平均寿命的流程图如图 2 所示。在求解可靠性时，只需要加入对系统寿命和求解时刻的大小判断，如果系统未失效，则计数值加一。

2.2 切换器的状态

切换器可能出现如图 3、4 的故障。对于切换器 A，g_{A0}表示其处于正常工作状态，而g_{A1}、g_{A2}、g_{A3}分别表示处于故障 A1、A2、A3 状态。同理，切换器 B 是一个 3 状态元件。

切换器 A 和 B 的使用寿命 T 的概率密度分布和各种故障概率如式 1-7 所示。**程序中，使用 exprnd 随机产生使用寿命$lifeA$、$lifeB$，再使用轮盘赌判定其故障的类型$stateA$、$stateB$。**

$$f_{T_A}(\tau)=\lambda_A e^{-\lambda_A \tau} 其中 \frac{1}{\lambda_A}=5.90\times10^4 hour \quad (1\text{-}7)$$

$$P_{EA1}=Pr(A1\ occurs \mid A\ is\ failed)=0.20$$
$$P_{EA2}=Pr(A2\ occurs \mid A\ is\ failed)=0.15$$
$$P_{EA3}=Pr(A3\ occurs \mid A\ is\ failed)=0.65$$
$$f_{T_B}(\tau)=\lambda_B e^{-\lambda_B \tau} 其中 \frac{1}{\lambda_B}=2.20\times10^5 hour$$
$$P_{EB1}=Pr(B1\ occurs \mid B\ is\ failed)=0.45$$
$$P_{EB2}=Pr(B2\ occurs \mid B\ is\ failed)=0.55$$

2.3 节点的状态

节点的状态由其内部的切换器状态的组合来确定，如表 1。其中g_{N0}表示节点性能完好，定义别名g_{PF}（意为 perfectly functioning）；g_{N1}表示只能作为从节点，别名g_{SO}（意为 slave only）；g_{N2}表示或者作为主节点，或者作为不阻塞总线的失效节点，别名g_{DM}（disable / master）；g_{N3}表示只能作为主节点，否则就会阻塞总线，别名g_{MO}（master only）；g_{N4}表示成为不阻塞总线的失效节点，别名g_{DN}（disable node）；g_{N5}表示节点总是阻塞总线，别名g_{FB}（failed bus）。

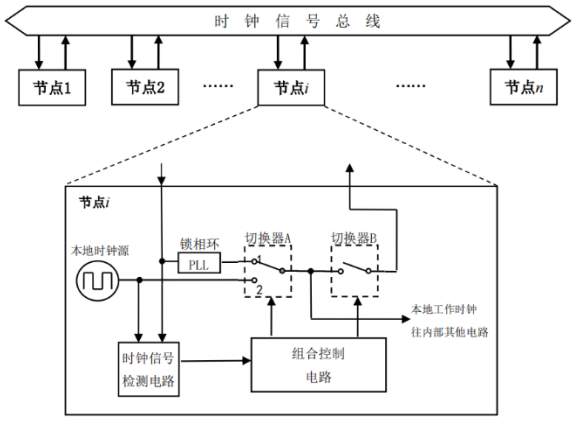

图 1 多节点声纳系统中时钟同步机制示意图[1]

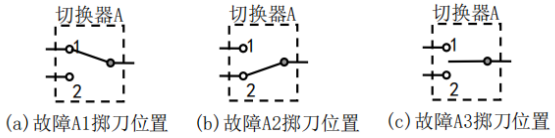

图 3 切换器 A 的故障类型[1]

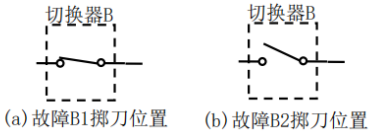

图 4 切换器 B 的故障类型[1]

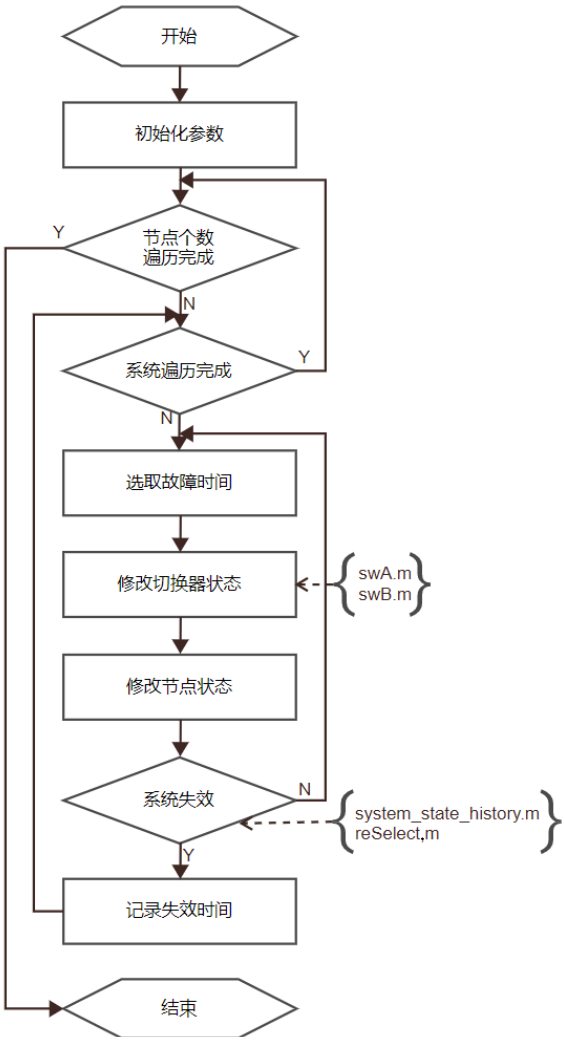

图 2 程序流程图

表 1 切换器-节点状态映射关系[1]

切换器 A 状态	切换器 B 状态	节点状态	切换器 A 状态	切换器 B 状态	节点状态
g_{A0}	g_{B0}	g_{PF}	g_{A2}	g_{B0}	g_{DM}
	g_{B1}	g_{MO}		g_{B1}	g_{MO}
	g_{B2}	g_{SO}		g_{B2}	g_{DN}
g_{A1}	g_{B0}	g_{SO}	g_{A3}	g_{B0}	g_{DN}
	g_{B1}	g_{FB}		g_{B1}	g_{DN}
	g_{B2}	g_{SO}		g_{B2}	g_{DN}

程序中，使用二维矩阵*node_state*，其中每行对应一种切换器 A 状态，每列对应一种切换器 B 状态，根据切换器状态确定节点状态*ns*。将六种节点状态对应的节点个数分别定义为 PF，SO，DM，MO，DN 和 FB。

2.4 系统的状态

系统的整体工作状态sys_state由所有的 n 个节点状态的组合来确定。

- C1：FB>=1
- C2：MO>=2
- C3：PF+MO+DM==0
- C4：PF+SO+1(MO+DM>0)<3
- C5：FB==0
- C6：MO==1&&PF+SO>=2
- C8：FB+MO==0
- C9：PF>=1&&DM>=1&&PF+SO==2
- C7：(MO==0&&PF==0&&DM>=1&&SO>=2)||(MO==0&&PF>=1&&PF+SO>=3)

当 C1、C2、C3、C4 至少有一个满足，则系统处于状态 1，即确定不能正常工作；当 C1～C4 都不满足，C5 满足且 C6 和 C7 中至少有一个满足，则系统处于状态 2，即确定正常工作。

对于涉及到**主节点重选机制**的状态 3 和 4 的判断，给出如下**两种建模方式：**

一. 简化模型

如果系统既不处在状态 1，也不处在状态 2，且 C8 和 C9 均满足，有 DM/(DM+PF)的概率进入状态 3，即恰好能正常工作，有 PF/(DM+PF)的概率进入状态 4，即恰好不能正常工作。

二. 历史遗留模型（简称历史模型）

在主程序中增加变量$master$记录当前主节点编号，**在当前主节点不能工作在主模式时，按照 DM 和 PF 的比例随机选择新的主节点。在出现新的 MO 节点时，此节点成为主节点。**因此，C8 和 C9 均满足时，主节点状态为 DM 进入状态 3，否则进入状态 4，即恰好不能正常工作。

理论分析两种模型，有两种差别：

一. 如果损坏的节点是从节点，之后满足 C8 和 C9 时，由于**没有触发**主节点重选，所以按照历史模型系统会失效。而按照简化模型，满足 C8 和 C9 时会**直接重选**主节点，系统仍有存活的可能性。这种情况下历史模型比简化模型更容易失效。

二. 在满足 C8 和 C9 时，每当**发生与主节点重选无关变化时**（SO->SO，作为主节点 PF->DM 等），按照简化模型，仍然会重选主节点，使系统可能失效。而历史模型会正确模拟，系统正常工作。这种情况下简化模型比历史模型更容易失效。

在下面的实验中，我测试了这两种差别谁是主要影响因素以及影响程度，并且权衡代码复杂性和模拟准确度给出了我选择的模型。

3 结果分析与模型改进

3.1 可靠性测试

系统可靠性定为系统工作寿命超过 30000 hours 的概率，用频率代替概率，采用蒙特卡洛方法，模拟 100000 套同型系统的运行状况。

表 2 可靠性测试

节点总数	3	4	5	6	7	8	9	10	11
简化模型 Rw	0.3289	0.6285	0.8028	0.8789	0.9084	0.9122	0.9040	0.8883	0.8764
历史模型 Rw	0.3352	0.6278	0.8013	0.8784	0.9080	0.9128	0.9027	0.8891	0.8749
节点总数	12	13	14	15	16	17	18	19	20
简化模型 Rw	0.8607	0.8429	0.8291	0.8079	0.7921	0.7741	0.7556	0.7384	0.7244
历史模型 Rw	0.8574	0.8438	0.8268	0.8094	0.7937	0.7748	0.7562	0.7396	0.7203

实验结果如表 2 所示。节点总数为 8 时，系统可靠性最大，简化模型为 0.9122，历史模型为

0.9128。对于简化模型，可靠性随节点总数变化如图 5 所示。

在前文的系统状态和相应条件上进一步分析，引起系统失效的原因大体可以分为三类：

一. 出现了阻塞总线的节点（C1 和 C2），记为bug_1。

二. 有效节点数不足 3 个（C4 和 sys4），分别记为bug_2和bug_3。

三. 主节点数量不足（C3），记为bug_4。

若同时出现多个 bug，按照$bug_2 > bug_3 > bug_4 > bug_1$的优先级判定。对于简化模型，在节点总数为 8 时再次实验，统计四种 bug 出现的比例如表 4 和图 7 所示。可知**阻塞总线为引起系统失效的主要原因**，占失效系统的 86.9%。同时也能看出，**简化模型中，由 sys4 引起的bug_3发生概率很少**，所以两个模型的**结果无明显差异**。

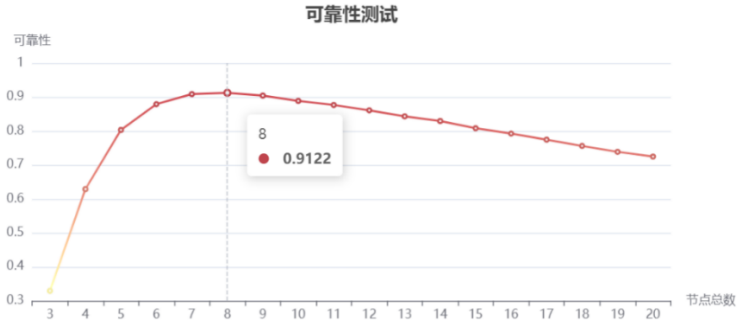

图 5 可靠性测试

3.2 平均工作寿命测试

首次失效时间，又称工作寿命，是系统从初始时间到首次发生失效的时间。我们可以将 S 套系统的首次失效时间求平均值，用来替代首次失效时间的期望。

表 3 平均工作寿命测试

节点总数	3	4	5	6	7	8
简化模型$E(T_f)/hour$	28674.135	51035.685	68533.874	81825.690	91120.871	98301.097
历史模型$E(T_f)/hour$	28585.055	51272.654	69185.898	83043.167	92663.104	99114.463
节点总数	9	10	11	12	13	14
简化模型$E(T_f)/hour$	102345.203	104114.649	104803.970	104459.807	102028.938	100004.064
历史模型$E(T_f)/hour$	102884.929	104870.332	105072.872	104167.504	102844.025	100250.090
节点总数	15	16	17	18	19	20
简化模型$E(T_f)/hour$	96947.873	93734.349	89986.919	86812.690	83118.476	79847.423
历史模型$E(T_f)/hour$	97389.129	94031.927	90451.586	86885.261	83367.089	79799.543

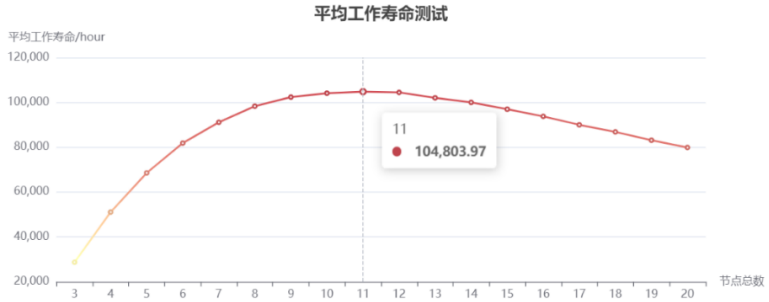

图 6 平均工作寿命测试

实验结果如表 3 和图 6 所示。节点总数为 11 时，系统平均工作寿命最大，简化模型为 104803.970 hours，历史模型为 105771.371 hours。对于简化模型，可靠性随节点总数变化如图 7 所示。

明显看出历史模型比简化模型的平均工作寿命更长，可见 2.4 节提到的情况二“发生与主节点重选无关变化”是主要影响因素。在节点总数为 11 时，简化模型相对历史模型的误差为 0.26%。可知**简化模型在大幅降低模型复杂度的基础上带来的误差可以接受**。

针对简化模型，统计节点总数为 11 时，四种 bug 出现的比例，如表 4 和图 8。其中 r 定义为系统到达最大工作时长仍然没有失效。**阻塞总线仍为引起系统失效的主要原因。**

表 4 引起系统失效的 bug 比例统计

测试参数	r （并未失效）	bug_1 （阻塞总线）	bug_2 （有效节点不足）	bug_3 （主节点不足）	bug_4 （主节点不足）
可靠性	0.9105	0.0778	0.0066	0.0048	0.0003
平均寿命	0.1717	0.5002	0.2018	0.0633	0.0631

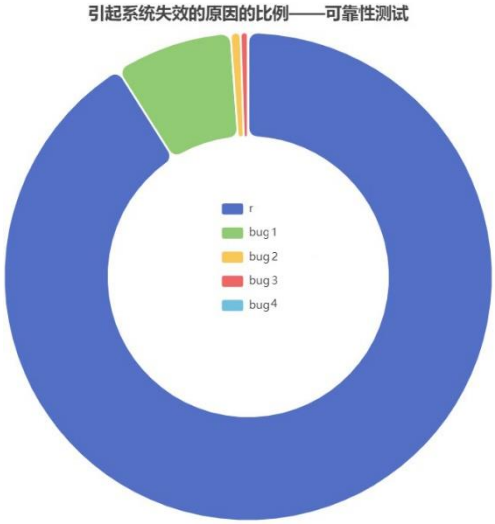

图 7 可靠性测试失效原因比例

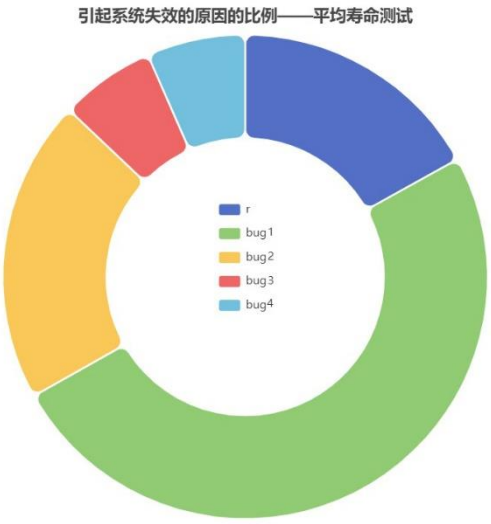

图 8 平均寿命测试失效原因比例

3.3 模型改进——双总线系统

由于阻塞总线为系统失效的主要原因，故可以采用双总线结构提高系统的可靠性和平均寿命。**当一条总线被阻塞后，所有的节点切换到另一条总线工作。只有当两条总线均被阻塞时，系统才失效。切换器状态和节点状态的判别不变。**由 3.2 知“历史遗留问题”带来的误差很小，故在简化模型的基础上拓展。

系统状态的判别在[1]的基础上进行如下修正。**条件均只需针对 FB 和 MO 进行修正，其余条件定义与单总线系统相同。**

- C1：FB>=2
- C2-1：FB==1&&MO>=2
- C2-2：FB==0&&MO>=3
- C8：MO==0&&FB<=1

当 C4 满足时，有效节点数不足 3 个，发生bug_1；当 C8 和 C9 均满足，有$\frac{PF}{PF+DM}$的概率有效节点数不足 3 个，发生bug_2；当 C3 满足时，无主节点，发生bug_3；当 C1、C2-1 和 C2-2 至少有一个满足，两条总线都被阻塞，发生bug_4；其余情况下，系统正常工作。实验结果如表 5、6 和图 9、10 所示。

表 5 双总线系统可靠性测试

节点总数	3	4	5	6	7	8	9	10	11
Rw	0.3289	0.6387	0.8270	0.9206	0.9657	0.9826	0.9875	0.9886	0.9862
节点总数	12	13	14	15	16	17	18	19	20
Rw	0.9835	0.9794	0.9753	0.9694	0.9643	0.9568	0.9521	0.9439	0.9362

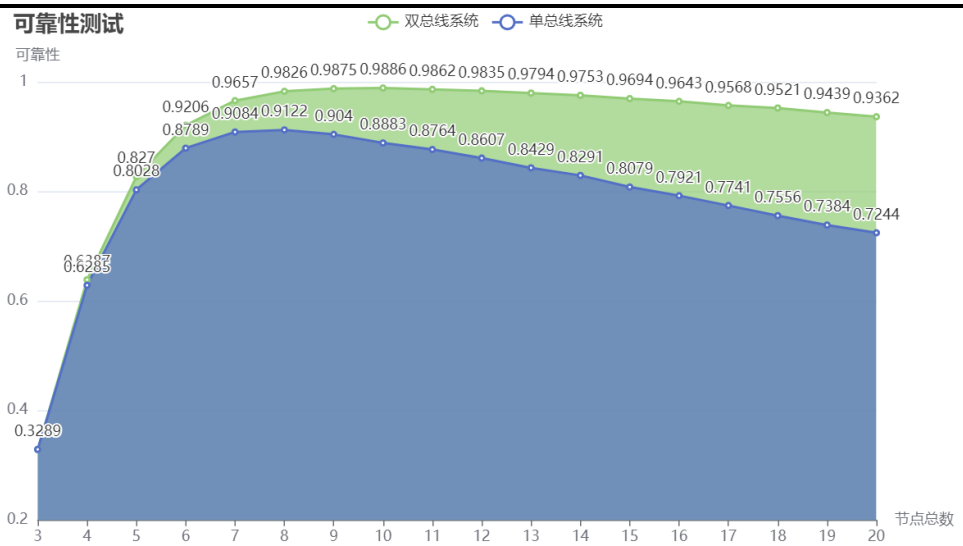

图 9 双总线系统可靠性测试

表 6 双总线系统平均工作寿命测试

节点总数	3	4	5	6	7	8
$E(T_f)/hour$	28663.651	52105.179	72199.265	89737.487	104647.195	116986.848
节点总数	9	10	11	12	13	14
$E(T_f)/hour$	126745.892	134115.243	139972.310	144254.858	146829.967	147790.725
节点总数	15	16	17	18	19	20
$E(T_f)/hour$	147937.232	147072.786	145385.843	143324.825	140506.110	137421.222

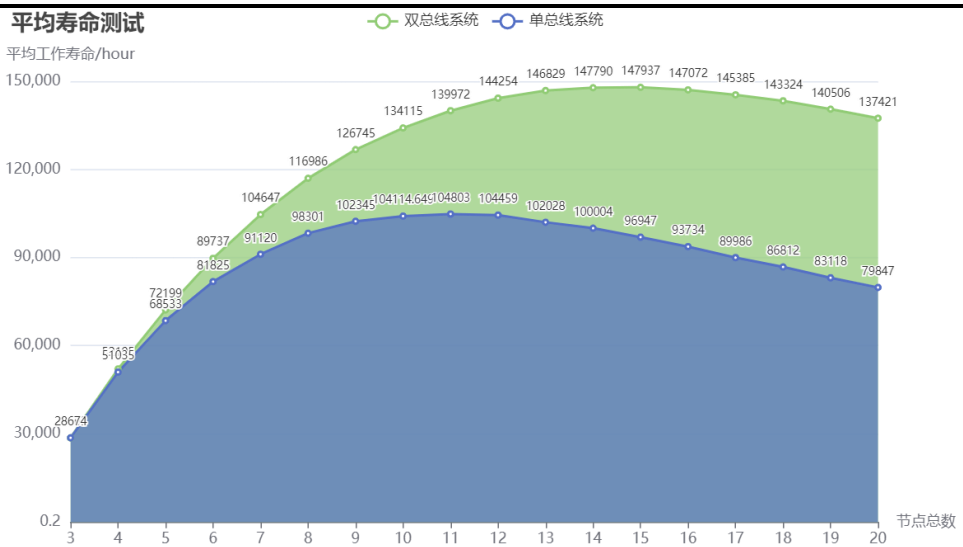

图 10 双总线系统平均寿命测试

随着节点总数的提高，系统失效的原因是阻塞总线的可能性提高，所以双总线对系统可靠性和平均寿命的提升更加明显。当使用双总线系统时，使用 10 和 15 个节点可分别使可靠性和平均寿命最大。

4 理论求解系统可用性

4.1 理论运算法

系统可用性是指系统在时刻 t，瞬时状态为正常工作状态的概率。由于系统会发生先失效而后又恢复功能的所谓“复活”现象，所以，可用性数值会比可靠性为高。而“复活”现象的发生概率比较小，所以用系统可用性计算结果作为可靠性近似解是合理的。下面通过数学推导得出系统可用性。

当系统处于 t 时刻，切换器 A、B 各个状态下的概率如式 8-14。

$$p_{A0}(t) = e^{-\lambda_A t}$$
$$p_{A1}(t) = 0.20\left(1 - e^{-\lambda_A t}\right) \quad (8\text{-}14)$$
$$p_{A2}(t) = 0.15\left(1 - e^{-\lambda_A t}\right)$$
$$p_{A3}(t) = 0.65\left(1 - e^{-\lambda_A t}\right)$$
$$p_{B0}(t) = e^{-\lambda_B t}$$
$$p_{B1}(t) = 0.45\left(1 - e^{-\lambda_B t}\right)$$
$$p_{B2}(t) = 0.55\left(1 - e^{-\lambda_B t}\right)$$

于是，我们可以确定节点的六种状态所对应的概率如式 15-20。

$$P_{PF}(t) = p_{A0}(t)p_{B0}(t)$$
$$P_{MO}(t) = p_{A0}(t)p_{B1}(t) + p_{A2}(t)p_{B1}(t) \quad (15\text{-}20)$$
$$P_{SO}(t) = p_{A0}(t)p_{B2}(t) + p_{A1}(t)p_{B0}(t) + p_{A1}(t)p_{B2}(t)$$
$$P_{FB}(t) = p_{A1}(t)p_{B1}(t)$$
$$P_{DM}(t) = p_{A2}(t)p_{B0}(t)$$
$$P_{DN}(t) = p_{A2}(t)p_{B2}(t) + p_{A3}(t)(p_{B0}(t) + p_{B1}(t) + p_{B2}(t))$$

通过各个节点的状态概率，我们可以进一步分析系统的状态概率如式 21。

$$A(t) = C_5(t) \times (C_6(t) + C_7(t)) + C_8(t) \times C_9(t) \times Pr(t) \quad (21)$$

其中各参量如式 22-27。

$$C_5(t) = (1 - P_{FB}(t))^n \quad (22\text{-}27)$$

$$C_6(t) = C_n^1 P_{MO}(t)\left(1 - P_{MO}(t)\right)^{n-1}\left(\sum_{i=2}^{n-1} C_{n-1}^i \left(P_{PF}(t) + P_{SO}(t)\right)^i \left(1 - P_{PF}(t) - P_{SO}(t)\right)^{n-1-i}\right)$$

$$C_7(t) = \left(1 - P_{MO}(t)\right)^n \left(1 - \left(1 - P_{PF}(t)\right)^n\right)\left(\sum_{i=3}^{n-1} C_{n-1}^i \left(P_{PF}(t) + P_{SO}(t)\right)^i \left(1 - P_{PF}(t) - P_{SO}(t)\right)^{n-1-i}\right)$$
$$+ \left(1 - P_{MO}(t) - P_{PF}(t)\right)^n \left(1 - \left(1 - P_{DM}(t)\right)^n\right)\left(\sum_{i=2}^{n-1} C_{n-1}^i P_{SO}(t)^i \left(1 - P_{SO}(t)\right)^{n-1-i}\right)$$

$$C_8(t) = (1 - P_{FB}(t) - P_{MO}(t))^n$$

$$C_9(t) = C_n^2(t)\left(P_{PF}(t) + P_{SO}(t)\right)^2 \left(1 - P_{PF}(t) - P_{SO}(t)\right)^{n-2} \left(1 - \left(1 - P_{PF}(t)\right)^n\right)\left(1 - \left(1 - P_{DM}(t)\right)^n\right)$$

$$Pr(t) = \frac{P_{DM}(t)}{P_{DM}(t) + P_{PF}(t)}$$

代入$t = w = 30000\ hours$可以获得 A(w)。计算量非常繁重，复杂度体现在式 22-26 的组合

数的求和运算，所以利用 Matlab 提出穷举法来替代。

4.2 穷举法

在计算出切换器概率和节点概率后，穷举每一种**节点组合**的概率，判断是否满足状态 2 和状态 3 的条件。如果满足，利用组合数计算当前节点组合出现的概率，再计算可用性。结果如表 7 所示。

表 7 穷举法求解系统可用性

节点总数	3	4	5	6	7	8	9	10	11
A	0.3361	0.6327	0.8049	0.8849	0.9142	0.9193	0.9134	0.9027	0.8898
历史模型 Rw	0.3352	0.6278	0.8013	0.8784	0.9080	0.9128	0.9027	0.8891	0.8749
节点总数	12	13	14	15	16	17	18	19	20
A	0.8759	0.8613	0.8463	0.8310	0.8155	0.7997	0.7839	0.7679	0.7519
历史模型 Rw	0.8574	0.8438	0.8268	0.8094	0.7937	0.7748	0.7562	0.7396	0.7203

可用性高于可靠性，是由于在两种特定的情况下，系统会发生先失效而后又恢复功能的现象：

一. MO 节点状态转为 DN，争抢总线的情况消失，系统可能恢复功能。

二. 系统处于 sys4 状态，某 DM 节点状态转为 MO，于是必然成为主节点，使系统内有效节点增加 1 个，系统功能得以恢复；或者主节点发生故障进入 SO 状态，一个 DM 节点成为主节点，使系统内有效节点增加 1 个，系统功能得以恢复。[1]

5 结论

节点总数为 8 时，系统可靠性最大，为 91.28%。节点总数为 11 时，系统平均工作寿命最大，为 105072.872 hours。考虑“历史遗留问题”的历史模型与[1]中提出的简化模型相比，失效概率降低，但由于简化模型误差在可接受的范围内，可以采用简化模型。在这两种情况下，引起系统失效的主要原因均为阻塞总线，故采用双总线系统作为优化模型。

随着节点总数的提高，双总线系统的优化效果更明显。在双总线系统中，节点总数为 10 时，系统可靠性最大，为 98.86%。节点总数为 15 时，系统平均工作寿命最大，为 147937.232 hours。

穷举法理论求解系统可用性略高于系统可靠性，证明了“复活”现象的存在。

6 参考文献

[1] 上海交通大学电子工程系. 工程问题建模与仿真之案例课题 2_V2.12 20220305[EB].

7 附录

7.1 单总线系统（历史模型）

main.mlx

```
clear

%% Initiation
S = 100000; % 系统数量
w = 30000; % 需要的有效工作时间
longest = 200000; % 首次失效时间最大值
% perfectly functioning: ns = 0
% slave only: ns = 1
% disable/master: ns = 2
% master only: ns = 3
% disable node: ns = 4
% failed bus: ns = 5
node_state = [0, 3, 1; % 切换器A为状态0
              1, 5, 1; % 切换器A为状态1
              2, 3, 4; % 切换器A为状态2
              4, 4, 4]; % 切换器A为状态3
r = zeros(1,20); % 可靠系统数量
R = zeros(1,20); % 可靠系统比例
mean_Tf = zeros(1,20); % 系统平均首次失效时间

%% Body
for point_num = 3:20 % 节点总数
    Tf = zeros(1,S); % 系统首次失效时间
    for s = 1:1:S
        %% Initiation
        lifeA = exprnd(59000,1,point_num); % 切换器A的使用寿命
        lifeB = exprnd(220000,1,point_num); % 切换器B的使用寿命
        stateA = zeros(1,point_num); % 切换器A的状态
        stateB = zeros(1,point_num); % 切换器B的状态
        ns = zeros(1,point_num); % 节点的状态
        master = 1; % 主节点标号

```

```
33          No_A_B = [(1:point_num),(1:point_num);ones(1,point_num),2*
                ones(1,point_num);lifeA,lifeB]'; % 第一列是序号，第二列中
                "1"表示swA，"2"表示swB，第三列是寿命
34          No_A_B = sortrows(No_A_B,3); % 对切换器按照寿命排序
35          No_A_B = No_A_B'; % 第一行是序号，第二行中"1"表示swA，
                "2"表示swB，第三行是寿命
36          No = No_A_B(1:2,:);
37          life_A_B = No_A_B(3,:);
38
39          %% Simulation
40          for i = 1:2*point_num
41              time = life_A_B(i); % 采用变步长，选取每一个时间
42              if time >= longest
43                  Tf(s) = longest;
44                  break
45              end
46              if No(2,i) == 1 % 切换器A发生故障
47                  stateA(No(1,i)) = swA;
48              else % 切换器B发生故障
49                  stateB(No(1,i)) = swB;
50              end
51              prev = ns(No(1,i)); % 当前故障节点之前的状态
52              ns(No(1,i)) = node_state(stateA(No(1,i))+1,stateB(No(1,i))
                    +1);
53              [sys_state,master] = system_state_history(ns,prev,No(1,i),
                    master); % 系统的状态
54              if sys_state==1 || sys_state==4
55                  Tf(s) = time;
56                  break;
57              end
58          end
59
60          %% Count
```

```
61            if time >= w
62                r(point_num) = r(point_num) + 1;
63            end
64
65            if i == 2 * point_num + 1
66                Tf(s) = longest;
67            end
68        end
69        R(point_num) = r(point_num) / S;
70        mean_Tf(point_num) = mean(Tf);
71    end
72
73    %% Display
74    R % 不同节点总数下的系统可靠性
75    [Rw,nice_point_num_R] = max(R) % 最大的系统可靠性&使系统可靠性最
          大的节点总数
76    mean_Tf % 不同节点总数下的平均工作寿命
77    [Tfw,nice_point_num_Tf] = max(mean_Tf) % 最大的平均工作寿命&使平
          均工作寿命最大的节点总数
```

swA.m

```
1   function [stateA] = swA
2       p = rand(1,1);
3       if p < 0.20
4           stateA = 1;
5       elseif p < 0.35
6           stateA = 2;
7       else
8           stateA = 3;
9       end
```

swB.m

```
1   function [stateB] = swB
```

```
    p = rand(1,1);
    if p < 0.45
        stateB = 1;
    else
        stateB = 2;
    end
```

system_state_history.m

```
% 根据各种状态节点数目来判断系统状态1-4以及确定主节点
function [sys_state,master] = system_state_history(ns,prev,i,master)
    sys_state = 2; % 状态2：系统确定能有效工作

    PFidx = find(ns==0); % PF的节点的下标
    DMidx = find(ns==2); % DM的节点的下标
    MOidx = find(ns==3); % MO的节点的下标

    PF = length(PFidx); % perfectly functioning的节点个数
    SO = length(find(ns==1)); % slave only的节点个数
    DM = length(DMidx); % disable/master的节点个数
    MO = length(MOidx); % master only的节点个数
    FB = length(find(ns==5)); % failed bus的节点个数

    % 判断系统是否处于状态1
    if MO + DM > 0
        ttmp = 1;
    else
        ttmp = 0;
    end
    if FB>=1 || MO>=2 || PF+MO+DM==0 || PF+SO+ttmp<3
        sys_state = 1; % 状态1：系统确定不能有效工作
    end

    % 在以下情况下需要进行主节点重选
```

```
26      if master == i && sys_state ~= 1
27          if prev == 0
28              if ns(i) == 1 || ns(i) == 4
29                  master = reSelect(PFidx,DMidx,PF,DM);
30              end
31          elseif (prev == 3 && ns(i) == 4) || (prev == 2 && ns(i) == 4)
32              master = reSelect(PFidx,DMidx,PF,DM);
33          end
34      end
35      if MO ~= 0
36          master = MOidx;
37      end
38
39      % 判断其他系统状态
40      if sys_state ~= 1 && FB+MO==0 && (PF>=1&&PF+SO==2&&
            DM>=1)
41          if ns(master) == 2
42              sys_state = 3; % 状态3: 系统恰能有效工作
43          else
44              sys_state = 4; % 状态4: 系统恰不能有效工作
45          end
46      end
```

reSelect.m

```
1   function master = reSelect(PFidx,DMidx,PF,DM)
2       randIDX = randi([1 PF+DM]);
3       allIDX = [PFidx DMidx];
4       master = allIDX(randIDX);
```

7.2 双总线系统

system_state_dual.m

```
% 根据各种状态节点数目来判断系统状态1-4
function [bug_state] = system_state_dual(PF,SO,DM,MO,FB)
    master = 0;
    if MO + DM > 0
        master = 1;
    end
    if PF + SO + master < 3
        bug_state = 1; % C4引起有效节点数不足3个
    elseif (MO==0&&FB<=1) && (PF>=1&&PF+SO==2&&DM>=1)
        pr = rand(1,1);
        if pr >= DM / (DM+PF)
            bug_state = 2; % sys4引起有效节点数不足3个
        else
            bug_state = 0;
        end
    elseif PF + MO + DM == 0
        bug_state = 3; % 主节点数量不足
    elseif FB >= 2 || (FB==1&&MO>=2) || (FB==0&&MO>=3)
        bug_state = 4; % 阻塞总线
    else
        bug_state = 0;
    end
```

7.3 穷举法求解可用性

main_theory.mlx

```
clear

k = 3;
w = 30000;
A = zeros(1,20); % 可用性

lambdaA = 1 / (5.9 * 10^4);
lambdaB = 1 / (2.2 * 10^5);

% 切换器状态概率
PA0 = exp(-lambdaA * w);
PA1 = 0.20 * (1 - PA0);
PA2 = 0.15 * (1 - PA0);
PA3 = 0.65 * (1 - PA0);
PB0 = exp(-lambdaB * w);
PB1 = 0.45 * (1 - PB0);
PB2 = 0.55 * (1 - PB0);

% 节点状态概率
P_PF = PA0 * PB0;
P_SO = PA0 * PB2 + PA1 * PB0 + PA1 * PB2;
P_DM = PA2 * PB0;
P_MO = PA0 * PB1 + PA2 * PB1;
P_DN = PA2 * PB2 + PA3 * (PB0 + PB1 + PB2);
P_FB = PA1 * PB1;

for n = 3:20
    for PF = 0:n
        for SO = 0:n-PF
            for DM = 0:n-PF-SO
                for MO = 0:n-PF-SO-DM
                    for DN = 0:n-PF-SO-DM-MO
```

```
33                          FB = n - PF - SO - DM - MO - DN;
34                          %状态2
35                          if FB==0 && ((MO==1&&PF+SO>=k-1) || ((
                                MO==0&&PF>=1&&PF+SO>=k)||(MO
                                ==0&&PF==0&&DM>=1&&SO>=k-1)))
36                              A(n) = A(n) + nchoosek(n,PF) * nchoosek(n-
                                    PF,SO) * nchoosek(n-PF-SO,DM) *
                                    nchoosek(n-PF-SO-DM,MO) *
                                    nchoosek(n-PF-SO-DM-MO,DN) * ...
37                                  P_PF^PF * P_SO^SO * P_DM^DM *
                                        P_MO^MO * P_DN^DN * P_FB^
                                        FB;
38                          %状态3
39                          elseif FB+MO==0 && (PF>=1&&PF+SO==k
                                -1&&DM>=1)
40                              A(n) = A(n) + nchoosek(n,PF) * nchoosek(n-
                                    PF,SO) * nchoosek(n-PF-SO,DM) *
                                    nchoosek(n-PF-SO-DM,MO) *
                                    nchoosek(n-PF-SO-DM-MO,DN) * ...
41                                  P_PF^PF * P_SO^SO * P_DM^DM *
                                        P_MO^MO * P_DN^DN * P_FB^
                                        FB * DM/(DM+PF);
42                          end
43                      end
44                  end
45              end
46          end
47      end
48      fprintf('节点数n = %d时,可用性A = %1.4f\n',n,A(n));
49  end
```