# 面向工控领域的拟态安全处理机架构

魏 帅, 于 洪, 顾泽宇, 张兴明

国家数字交换系统工程技术研究中心 郑州 中国 450002

**摘要** 近年来, 针对工控系统的攻击越来越多, 工业控制系统应用大多涉及国计民生, 其安全问题不容忽视。我国工控系统中现场 PLC、终端、RTU 等控制设备大部分使用国外的控制组件, 对于未知的逻辑炸弹和后门基本没有安全防护能力。为此, 本文以拟态技术为基础, 提出了一种通用的拟态安全处理机架构, 采用基于状态保存的两步清洗技术和高可靠判决策略, 使得符合该架构规范的应用程序均能借助拟态处理架构防护操作系统、处理机和外围器件可能出现已知或未知的后门/漏洞, 最后的仿真验证结果验证了该系统可以有效地抵御多种类型的攻击。

**关键词** 拟态安全; 工业控制; 处理机; 清洗和判决方法
**中图法分类号** TP309 DOI 号 10.19363/j.cnki.cn10-1380/tn.2017.01.005

# Architecture of Mimic Security Processor for Industry Control System

WEI Shuai, YU Hong, GU Zeyu, ZHANG Xingming

National Digital Switching System Engineering & Technological R&D Center, Zhengzhou 450002, China

**Abstract** Over the last years, attacks targeting ICSs, most of which largely concern national welfare and the people's livelihood, are prevailing and should not be neglected. In our country, the vital pieces in ICSs such as PLCs, terminals, RTUs and so on mostly come from abroad. Without independency or control over these pieces, we cannot achieve protection against unknown logic bombs or backdoors. Based on Mimic Technology, this paper presents a new general architecture of mimic security processor. This processor uses status-based two-step cleanout method and high-reliable arbitration method. APPs that conform to this architecture can protect the operating system, the processor and other modules from known or unknown backdoors/vulnerabilities. The simulations at the end prove mimic security processor can defend multi types of attacks.

**Key words** Mimic security; industry control; processor; cleanout and arbitration method

## 1 引言

近年来, 工控安全事件频发。据美国工业控制系统网络应急小组(ICS-CERT)的报告显示, 工控安全事件 2009 年发生 9 起, 2010 年发生 41 起, 2011 年和 2012 年共发生 198 起, 2013 年发生 200 多起。国际典型案例有 2003 年 Davis-Besse 核电站因 Slammer 蠕虫病毒入侵导致系统计算机停机 6 小时以上; 2007 年美国爱达荷国家实验室一台为 13.8KV 网格测试台供电的柴油发电机因网络攻击而被毁; 2008 年 Hatch 核电厂某工程师在数据采集服务器上测试软件更新时, 在其不知情的情况下, 测试值被供应商软件同步设置到生产系统上, 结果导致反应堆自动停机事故; 2010 年 9 月伊朗"震网"事件, 致使布什尔核电站 1000 多台离心机瘫痪; 2014 年 12 月, 德国一家钢铁厂遭受高级持续性威胁(APT)攻击并造成重大物理伤害; 2015 年, 乌克兰某变电站控制系统持续遭到网络攻击, 至少三个区域的约 140 万居民的电力供应受到影响。在国内, 据文献[1]显示, 仅 2015 年, 就有多起包括电力公司、石化企业、钢厂等大型企业在内的控制网络感染网络病毒的事件发生。另外, 国内著名网络摄像头生产制造企业海康威视遭

**通讯作者:** 于洪, 硕士, 助理研究员, Email: yuhong_3210@163.com。

本课题得到上海市科研计划项目工业控制拟态安全处理器原型验证(14DZ1104800); 5G 大规模协作无线传输关键技术研发(2014AA01A704); 网络空间拟态安全异构冗余机制研究(No.61572520); 网络空间拟态防御基础理论研究(No.61521003); 国家自然科学基金面上项目(61572520)资助。

收稿日期: 2016-9-4; 修改日期: 2016-9-25; 定稿日期: 2016-12-22

遇“黑天鹅”安全门事件, 部分监控设备已被境外 IP 地址控制。种种事件表明, 一直以来被认为相对安全的工业控制系统已经成为黑客攻击的目标。

在我国, 工业控制系统的应用有超过 80%为涉及国计民生的关键基础设施(如铁路、城市轨道交通、供水、供电等), 其安全性涉及的不再是信息泄露等“小问题”, 而是关系生产安全和和经济发展, 会对现实世界造成直接影响的“大问题”。且对工控系统的攻击成本在不断降低, 某些攻击不需要利用复杂的攻击手段, 不需要还原业务系统运行过程, 就可以达到对工控系统攻击的目的。

越来越严峻的工控安全形势, 也引发了越来越多对工控安全的关注。在国家层面, 工信部于 2011 年发布了《关于加强工业控制系统信息安全管理的通知》。在此之后, 国内各行各业都对工控系统安全的认识达到了一个新的高度, 电力、石化、制造、烟草等多个行业陆续制定了相应的指导性文件。国家标准相关的组织 TC260、TC124 等也已经启动了相应标准的研究制定工作。在行业层面, 国内的科研院所、工控系统厂商和安全系统厂商以及一些工控安全方向的新兴企业都积极投入到工控安全的研究及产品研发当中[2]。2014 年以前与工控安全相关的产品主要是工业防火墙和工业隔离网关等以边界防护为主的产品, 目前主流产品包括检测类、防护类和检测预警类产品, 开始对工控系统全生命周期提供安全保障。

但是, 目前的所有工控安全产品仍然存在较大的防护空当, 只能防护传统的已知攻击, 对于未知攻击毫无办法。我国工控系统中现场 PLC、终端、RTU 等控制设备大部分使用国外的控制组件, 未实现自主可控, 对于未知的逻辑炸弹和后门基本没有安全防护能力。

本文以拟态技术为基础, 提出了一种基于非相似余度的拟态安全处理机架构。目标是为了构建一个通用的拟态安全处理机架构, 使得符合该架构规范的应用程序均能借助拟态处理架构防护操作系统、处理机和外围器件可能出现已知或未知的后门/漏洞。仿真验证结果表明该架构能够有效抵御多种类型的攻击, 提升系统的安全性。

本文首先分析了国内外研究现状, 介绍了系统的总体架构及采用三处理机异构冗余系统的依据, 接着分析了拟态调度器的状态转移图, 以及基于状态保存的两步清洗方式, 第四章介绍了高可靠调度器判决策略, 第五章对拟态安全处理机系统进行了仿真验证, 最后对拟态安全处理及架构进行了总结和展望。

## 2　国内外研究现状

在传统的飞机等可靠性比较高的设备上, 为了避免单台电子控制设备出现故障, 往往采取多余度设计, 但是这种设计大多是同构的[3,4,5,6], 也有部分采用异构设计[7,8,9,10], 但通常都会采用紧耦合的设计方式, 对同步要求较高[11,12,13,14], 其目的是为了提升系统的可靠性, 而非安全性。

刘志颖[15]等针对传统的 PLC 三重冗余系统存在的共因失效问题, 提出了一种异构的 PLC 三重冗余系统, 采用 OMRON 的 CS1D、SIEMENS 的 S7-400 和 ROCKWELL 的 Controllogix 三个不同的 PLC 构成主处理机的三重化冗余, 利用表决和表决自适应机制对冗余数据进行处理, 并设计了一套适用于异构冗余的软硬件结合的检测机制。但是该种异构冗余架构是紧耦合的, 内部功能逻辑时序相同, PLC 之间存在着严格的同步, 通用性受限。

满梦华等[16]在一种新型计算机系统结构框架的基础上, 提出了多核异构冗余模型, 设计了互关总线协议, 实现了相应的容错策略。为此, 设计了一种新的总线格式, 处理机之间通过总线时钟进行同步, 通过总线协议进行处理机之间的数据仲裁。这种实现同样要求 CPU 采样总线的时钟进行同步, 且总线设计代价较高, 处理机也受制于总线的带宽和仲裁, 使用范围受限。

本文拟在避免对上层软件和底层处理架构进行较大改动的基础上, 构建一个通用的拟态安全处理机架构。为此, 将拟态边界设置在 IO 接口(如 PCIE 和 CAN)处, 通过对 IO 接口处的输出数据进行仲裁比较, 可以将拟态的防护范围扩展至处理机之外, 包括操作系统、中间件等。这种方法相比较更加通用, 防护边界更大。并且根据拟态安全思想, 异构冗余系统之间尽量避免协同, 所以本文提出的拟态安全处理机架构各处理机之间没有相互之间的同步控制, 仅有的同步通过调度器实现。而前面几种异构冗余系统只是进行系统可靠性的考虑, 普遍采用了系统之间的同步控制。

事实上, 为了应对新时期下的网络安全威胁, 美国等发达国家在对国家安全防御进行了深入研究, 发现传统的网络防护方式根本不能抵御新的网络威胁, 传统防御已经过时, 应该从最初的“积极防御”向“重塑网络空间”的“引领型”安全观转变。移动目标防御(Moving Target Defense: MTD)技术作为美国近年来提出的网络空间“改变游戏规则”的革命性技术之

一, 受到了美国白宫的高度重视, 反映了美国网络安全防御技术的彻底转型[17]。

移动目标防御完全不同于以往的网络安全研究思路。它并不追求建立一种完美无瑕的系统来对抗攻击, 相反, 移动目标防御的思路是: “构建、评价和部署机制及策略是多样的、不断变化的。这种不断变化的思路可以增加攻击者的攻击难度及代价, 有效限制脆弱性暴露及被攻击的机会, 提高系统的弹性”。其含义就是通过移动要保护的对象来达到防护目标的技术[18]。而拟态安全处理机架构可以通过调度器的随机调度来实现系统的多样性和不断变化, 是移动目标防御的扩展。

拟态安全思想来源于拟态计算。针对目前计算机系统结构固定, 难以有效处理复杂应用的问题, 信息工程大学在国家 863 计划“新概念高效能计算机体系结构及系统研究开发”项目中提出了拟态计算, 核心是一种基于认知的主动重构计算机体系结构(PRCA)[19, 20]。在 PRCA 中, 涉及到元结构和应用结构。应用结构是一个特定应用实现高效计算处理的结构方案, 它是一种动态结构; 元结构是为生成一簇面向领域应用而提供的由所需的资源组成的基础结构, 是构成功能结构的基本单元。在 PRCA 中, 当元结构中的资源未被应用结构采用时, 应处于休眠或关闭状态, 以降低能耗。

PRCA 的基本思想是: 系统识别应用的需求和变化, 感知可利用的处理资源, 针对应用既定目标, 构建出适合于应用需求的应用处理结构, 并且该结构随着应用的变化, 如计算进展阶段、处理负荷等的变化, 而进行结构的主动变化, 达到“应用决定结构, 结构决定效能”的目的。

“拟态变换”(函数化)的体系结构是获得高效能计算的本质。实践中硬系统的结构变换受器件技术、实现复杂性、结构变换开销和价格因素的限制不可能做到任意变化。拟态计算通过分析高效能计算的需求, 提出并实现了基于认知的决策支持系统 PDSS、可重构互联网络 iBT、高阶可重构处理机 HRPU 和智能混合存储等一系列关键技术。当应用以安全为主要目标时, 拟态计算需要分析安全计算的需求, 研究通过拟态计算实现安全的机制。拟态计算是拟态安全的基础, 拟态变换是两者共同的技术要点, 但应用目的不同, 前者是通过拟态变换追求高效能的计算, 而后者则是通过拟态变换实现系统的安全可控。

相比于传统的计算或处理系统, 动态变结构计算系统具有天然的非持续性、非相似性和非确定性的基本属性。这与基于漏洞或后门的恶意攻击所依赖的传统系统的静态性、相似性和确定性正好相左, 攻击链要求的稳定计算环境在拟态计算系统中似乎表现为是随机变化的, 其中的软硬件漏洞或后门都变得很难被利用, 于是防御方基于拟态计算架构并引入其他的动态化和随机化措施, 不难构建起非对称的基于拟态安全的主动防御系统, 以应对网络空间的现实威胁, 达成系统安全可控的目的。

## 3 拟态处理机总体设计

### 3.1 拟态防护边界与攻击模型

在本文提出的拟态系统中, 拟态防护边界包括 CPU、驱动程序、操作系统以及运行在其上的中间件(如 DDS、WEB 等)。拟态处理机架构主要针对拟态防护边界内存在的漏洞与后门进行安全防护。

针对拟态处理机的攻击点主要集中 CPU、驱动程序、操作系统及中间件中存在的漏洞/后门, 这些漏洞/后门基本上采用预置方式, 既可能是设计者有意为之, 也可能是由于设计的复杂性造成的缺陷。以 CPU 为例, 如今 IBM power 系列 CPU 的代码量已达到 4000 多万行, INTEL 最新 CPU 的代码量也达到近 6000 万行, 其中存在设计缺陷在所难免, 因设计分工而引入的陷门也不可能彻底杜绝, 主动隐匿复杂的后门功能也有充分的物质基础。

此外, 由于工业控制系统的特殊性, 其中的信息部件可能较当代工业水平滞后很多。比如飞机、舰船、火炮的服役年限可能长达数十年, 其中某些装备的 CPU、操作系统等还是三十多年前的 80386、DOS 系统等。这些当时可能认为安全的软硬件经过时间的推移和广泛的应用变得不那么可靠, 这就给工控系统带来了较大的安全风险。

攻击模型方面, 由于工业控制系统通常是封闭或半封闭系统, 极少情况下与外界通信, 不会像互联网上的主机那样随时都可能遭受漏洞/后门被触发的风险, 但也存在着安全隐患, 总体来说, 有以下两种攻击方式:

1) 当安全等级较低的调试设备或者已被病毒感染的上位机等设备接入工控系统时, 这些调试机或者上位机中可能存在某些恶意程序, 因为有了和工控系统连接的通道(如以太网等), 就可以实施常用的攻击方法。常用的攻击方法包括病毒感染, 植入木马或恶意软件等。虽然攻击方式和种类千差万别, 但根据攻击者与系统的交互过程, 可分为扫描探测, 系统突破等。

• 扫描探测是攻击的发起环节, 攻击者通过探测来获取能够直接反映目标位置的系统信息、目标系统的软硬件构成等信息, 判断搜寻目标可能存在的漏洞。

• 系统突破是指通过代码注入或者篡改系统的各项配置参数或敏感数据的基础上, 实现对系统的持久控制, 通常是以木马、病毒或后门的形式驻留在系统中, 也可以通过隐蔽添加系统各种类别用户、修改用户权限等方式来进行控制。

2) 在使用非自主可控系统如 CPU、操作系统和软件系统时, 系统中可能存在可被攻击者利用的漏洞, 甚至被预埋、被植入后门程序, 例如攻击者可以在系统中预埋程序, 这些程序会监听 wifi 或者 GPS 信号, 当其检测到特殊的 wifi 或者 GPS 数据时, 就会触发故障, 使得系统无法正常工作。

### 3.2　总体架构

在网络空间中基于静态系统架构的攻击要比防护容易得多, 攻击者只需要发现一个可利用的漏洞就可以轻易侵入系统实施攻击。拟态安全处理机的核心就是通过配置冗余资源、加入随机策略来增加了系统的动态性、不确定性和不可预测性。同时, 在硬件层次加入异构冗余设计, 可以屏蔽未知硬件漏洞和后门带来的隐藏风险, 这是相对于其他防护手段先进的地方。

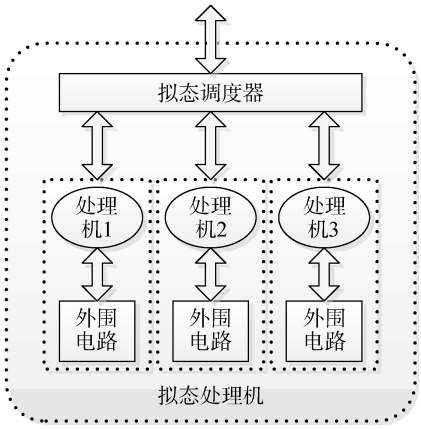



**图 1　拟态处理机总体架构**

拟态安全处理机一个完整的计算系统, 包括硬件和软件。硬件部分由三套异构冗余处理机及其外围电路以及一个拟态调度器组成, 其中异构冗余处理机负责执行任务, 调度器负责管理各处理部件并提供统一的对外接口。软件部分包括设备驱动、中间件以及异构操作系统。拟态安全处理机具有能同时提高系统传统可靠性和系统抵御操作系统层面、处理器硬件层面未知漏洞/后门能力的特点。

拟态调度器主要功能包括:

1) 用户与异构处理机之间的统一接口, 调度管理用户请求、仲裁异构处理机响应;

2) 伺服受控单元与异构处理机之间的统一接口, 调度管理伺服受控单元请求, 仲裁异构处理机响应;

3) 拟态处理机系统清洗管理及协同。

拟态调度器是拟态安全处理机的核心, 可采用自主设计的 ASIC 芯片实现, 在原理验证阶段可以采用 FPGA 进行仿真模拟。拟态调度器作为统一接口, 需要进行数据处理, 包括上行数据处理(用户单元/伺服受控单元至处理机)和下行数据处理(处理机至用户单元/伺服受控单元)。上行数据处理主要完成数据的分发; 下行数据处理主要完成数据类型判别、数据包提取和判决输出。

在异构多核处理的背景下, 单个处理机可能由于内部故障或者外部攻击输出错误结果, 此时, 需要调度器对各处理机的输出结果进行比较并输出调度器认为正确的结果。这个过程即称为判决。

数据清洗是指如果某个处理机发生故障如何消除错误并使其恢复正常。在处理机运行过程中, 单个处理机可能由于内部故障或者外部攻击输出错误结果。当处理机累积一定错误达到清洗条件时, 对处理机进行清洗。由清洗控制模块完成处理机清洗复位, 之后调度器会通过状态数据的交互使清洗后的处理机实现和其他处理机同步。

### 3.3　安全可靠性分析

对于多个异构处理机的输出结果, 拟态调度器需要进行比较输出。比较输出的过程称为“判决”。判决的方式有很多种, 包括随机判决输出、轮询判决输出、择多判决输出等[21]。可信度最高、最可靠的判决方式是择多判决, 通过比较在相同输入下处理机的输出, 发现异常的结果。择多判决利用了“多数行为准则”, 认为占多数的输出结果具有更高可信度, 占少数的输出结果为异常结果。虽然, 异常的输出结果不一定是遭受攻击的结果, “多数行为”也不一定就是正常状态下的行为, 但采用择多判决相对于单个处理机的情况, 仍然大大降低了输出错误结果的概率。

从理论上说, 采用的异构处理机越多, 获得的安全增益越大, 同时系统也越复杂, 成本等代价也越高, 需要在安全性和复杂性之间折衷。在本文的设计方案中, 采用的是三核异构处理机。下面通过三核异构处理机和四核异构处理机在输出结果错误率上的比较阐述采用三核处理机的理由。

#### 3.3.1　三核处理机安全增益

假设单个处理机出现正确结果的概率为 $p$, 出现错误结果(即被成功攻击)的概率为 $q$, 其中

$p+q=1$。以三核处理机为例，待判决的结果一共有如表 1 所示的几种情况:

**表 1　择多策略判断情况统计**

| 待判决结果 | | | 出现概率 | 判决 |
|---|---|---|---|---|
| 3 个结果正确 | | | $C_3^3\times p^3$ | 正确 |
| 2 个结果正确 | | | $C_3^2\times p^2\times q$ | 正确 |
| 1 个结果正确 | 2 个错误结果相同 | | $C_3^1\times p\times q^2\times\theta$ | 错误 |
| | 2 个错误结果不相同 | 选中正确结果 | $C_3^1\times p\times q^2\times(1-\theta)\times\theta'$ | 正确 |
| | | 选中错误结果 | $C_3^1\times p\times q^2\times(1-\theta)\times(1-\theta')$ | 错误 |
| 0 个结果正确 | | | $C_3^3\times q^3$ | 错误 |

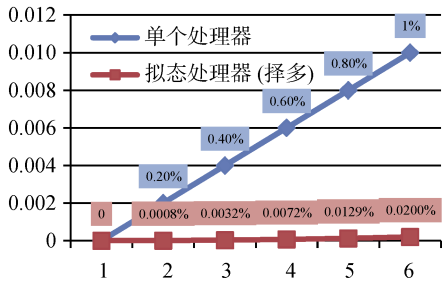
其中判决出现错误的概率总共为:

$$P_{\text{错误}}=C_3^1\times p\times q^2\times\theta+C_3^1\times p\times q^2\times(1-\theta)\times(1-\theta')+C_3^3\times q^3$$

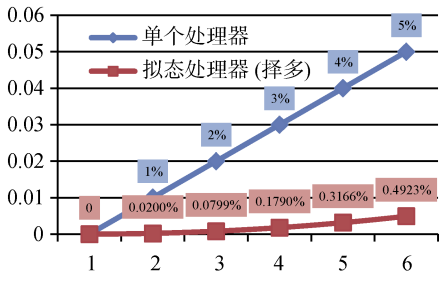
其中，$\theta$ 表示出现两个错误结果相同的概率；$\theta'$ 表示当三个待判决结果都不同时，选中正确结果作为输出的概率。

假设 $\theta=0.01$，$\theta'=0.33$ 时，单个处理机输出错误结果的概率 $q$ 变化范围为[0, 0.05]，拟态处理机输出错误结果 $P_{\text{错误}}$ 的概率变化如图 2 所示。

图 2(a)显示了在单个处理机错误率 $q\in[0, 0.01]$时，三核拟态处理机择多判决结果的错误率变化情况。




(a) $q\in[0,0.01]$的三核结果错误率对比




(b) $q\in[0,0.05]$的三核结果错误率对比

**图 2　三核结果错误率对比**

图 2(b)显示了在单个处理机错误率 $q\in[0, 0.05]$时，使用三核拟态处理机对输出结果的错误率的影响。

可以看出，在 $q\in[0, 0.05]$时，择多判决策略将处理机的结果错误率至少降低了一个数量级。而 $q\in[0, 0.01]$时，处理机结果错误率的降低更加明显，甚至超过了两个数量级。

### 3.3.2　四核处理机安全增益

如果采用四核异构处理机，则其待判决结果会出现表 2 所示的情况。

**表 2　择多策略判断情况统计**

| 待判决结果 | | | 出现概率 | 判决 |
|---|---|---|---|---|
| 4 个结果正确 | | | $C_4^4\times p^4$ | 正确 |
| 3 个结果正确 | | | $C_4^3\times p^3\times q$ | 正确 |
| 2 个结果正确 | 2 个错误结果相同 | 选中正确结果 | $C_4^2\times p^2\times q^2\times\theta\times\theta'$ | 正确 |
| | | 选中错误结果 | $C_4^2\times p^2\times q^2\times\theta\times(1-\theta')$ | 错误 |
| | 2 个错误结果不同 | | $C_4^2\times p^2\times q^2\times(1-\theta)$ | 正确 |
| 1 个结果正确 | 3 个错误结果相同 | | 极小概率，暂不考虑 | 错误 |
| | 2 个错误结果相同 | | $C_4^1\times p\times q^3\times\theta$ | 错误 |
| | 错误结果都不相同 | 选中正确结果 | $C_4^1\times p\times q^3\times(1-\theta)\times\theta''$ | 正确 |
| | | 选中错误结果 | $C_4^1\times p\times q^3\times(1-\theta)\times(1-\theta'')$ | 错误 |
| 0 个结果正确 | | | $C_4^4\times q^4$ | 错误 |

其中判决出现错误的概率总共为:

$$P_{\text{错误}}=C_4^2p^2q^2\theta(1-\theta')+C_4^1pq^3\theta+C_4^1pq^3(1-\theta)(1-\theta'')+C_4^4\times q^4$$

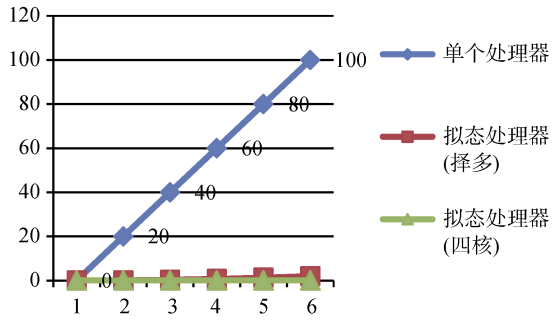
其中，$\theta$ 表示出现两个错误结果相同的概率；$\theta'$ 表示从两对结果中选出正确结果的概率，$\theta''$ 表示四个待判决结果都不同时，选中正确结果作为输出的概率。

假设 $\theta=0.01$，$\theta'=0.5$，$\theta''=0.25$ 时，单个处理机输出错误结果的概率变化范围为[0, 0.05]，三核及四核拟态处理机输出错误结果的概率变化如图 3 所示(其中单个处理机数据标签表示的是错误概率为万分之几，如数据标签为 100，则表示错误概率为 100/10000，即 1%):

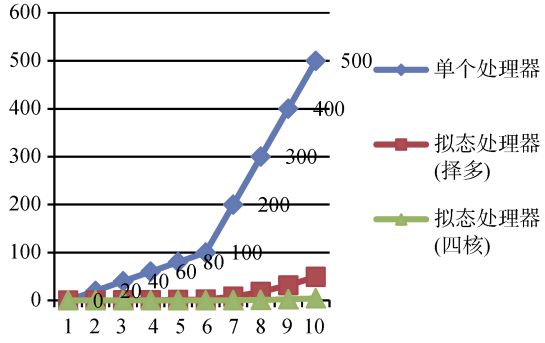
图 3(a)显示了在单个处理机错误率 $q\in[0, 0.01]$ 时，三核拟态处理机和四核拟态处理机择多判决结果的错误率比较情况。

图 3(b)显示了在单个处理机错误率 $q\in[0, 0.05]$ 时，三核拟态处理机和四核拟态处理机择多判决结果的错误率比较情况。

从图 3 可以看出，当 $q\in[0, 0.01]$ 时，使用四核处理机，其错误率降低幅度相比于三核处理机非常小，几乎可以忽略不计。在 $q\in[0.01,0.05]$ 范围内，略有差别。




(a) $q\in[0,0.01]$的四核结果错误率对比




(b) $q\in[0,0.05]$的四核结果错误率对比

**图 3　四核结果错误率对比**

### 3.3.3　总结

在错误率降低幅度差别不大的情况下，四核拟态处理机的设计比三核拟态处理机的设计更为复杂。硬件设计上，要增加控制部件，如功耗、体积、占用物理空间等；机制设计上，判决、调度和清洗机制也将更加复杂。使用四核设计将额外耗费显著的资源却并没有获得显著的安全增益。

另外，采用二核拟态处理机设计方案，使用上述的分析手段，可以得出二核拟态处理机中出现判决错误的概率为：$P_{\text{错误}}=q^2+C_2^1pq(1-\theta)$。其中，$\theta$ 表示当两个处理机结果不同时，选中正确结果的概率。在不引入可信度的情况下，$\theta=0.5$，则简化后有：$P_{\text{错误}}=q$。即二核拟态处理机相比单个处理机，错误率相等，无有效改善。

因此，采用三核设计是一个较好的折中方案。

## 3.4　拟态调度器状态机

拟态处理机采用异构冗余设计，从启动到任务执行完毕关机，每个异构处理机都会经历一系列的状态转换。总的来说，一个处理机可能出现的状态有：正常工作(A)、异常工作(B/C)、清洗中(W)、关闭中、未启动(S)。处理机状态发生改变时，会通知拟态调度器，拟态调度器由此感知每个处理机的状态。各处理机不同状态的组合构成了拟态调度器的状态。拟态调度器的状态主要包括启动(0)、三级安全工作模式(3-1)、三级安全清洗模式一(3-2)、三级安全清洗模式二(3-3)、二级安全工作模式(2-1)、二级安全清洗模式(2-2)和一级安全工作模式(1)。

本节主要描述了拟态调度器状态机，详细分析了每个状态出现的条件及对应的操作。

### 3.4.1　系统启动(0)

系统上电开始，拟态调度器进入系统启动状态，如图 4 所示。在这个状态下，拟态调度器负责接收由处理机发送的 Ready 信号(代表处理机启动成功)。如果接收到全部三个 Ready 信号，则进入一级安全工作模式；如果不是，且发生超时，则根据接收到的 Ready 信号个数，分别进入二级安全工作模式，或者三级安全工作模式；如果一个 Ready 信号都没有接收到，则启动失败。

### 3.4.2　三级安全工作模式(3-1)

在系统启动时，经过一段时间的等待(超时)，调度器发现只收到一个 Ready 信号，就会进入三级安全工作模式，如图 5 所示。在这个模式下，只有一个处理机在工作，属于安全等级最低的工作模式。另外两个处理机处于未启动或启动不成功状态，调度器不予理会。在这个模式下工作，调度器不会转移到更高的安全等级，只会在接收到关机信号

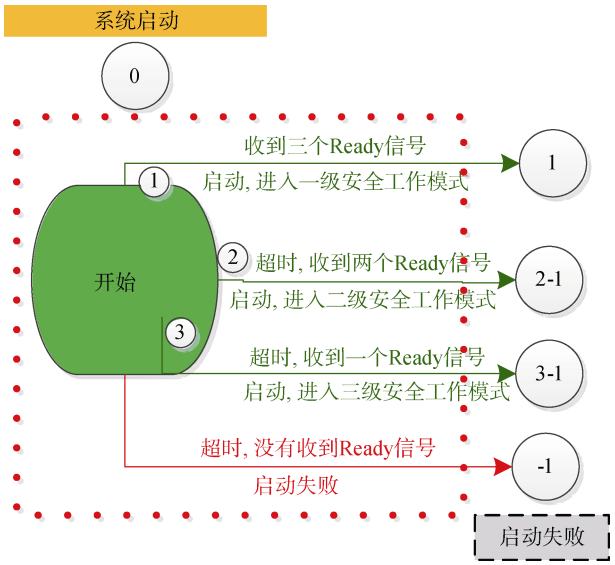



图 4 拟态调度器状态机总图

后进行关机。

### 3.4.3 三级安全清洗模式一(3-2)

在三级安全清洗模式一中, 三个处理机的状态分别为工作、清洗和未启动, 如图 6 所示。这个状态是从具有较高安全等级的二级安全工作模式转移来的, 如果清洗中的处理机完成清洗工作, 还会回到二级安全工作模式, 恢复到较高安全等级。

### 3.4.4 三级安全清洗模式二(3-3)

在三级安全清洗模式二中, 三个处理机的状态分别为工作、清洗和清洗, 即同时有两个处理机处于清洗状态中, 如图 7 所示。这个状态是从具有更高安全等级的二级安全清洗模式转换来的。当处理机完成清洗, 会恢复到二级安全清洗模式。如果两个清洗中的处理机同时完成清洗, 调度器也不会同时将两个处理机加入到工作中, 而是依次处理。

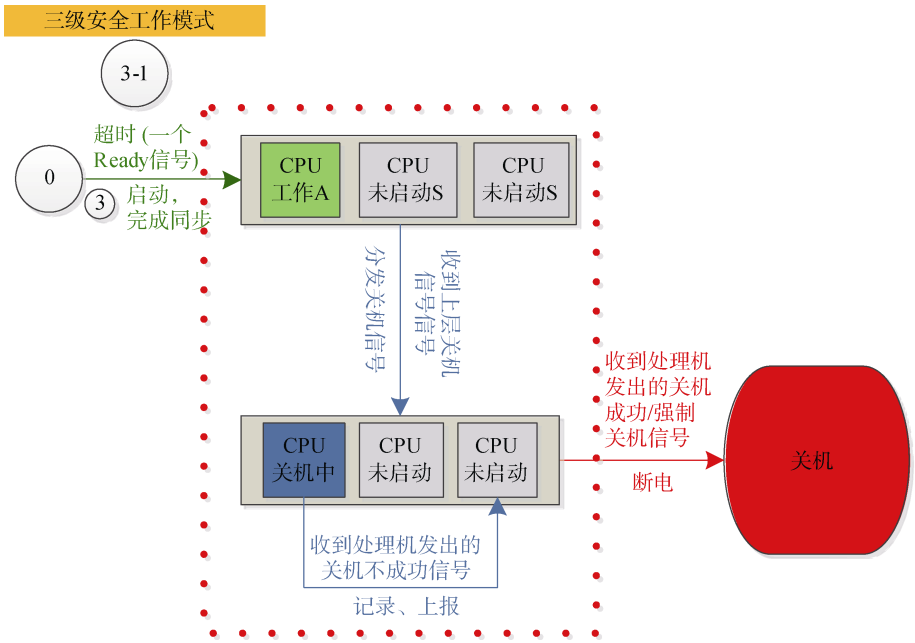



图 5 三级安全工作模式

### 3.4.5 二级安全工作模式(2-1)

在系统启动阶段, 出现系统超时后, 如果调度器收到两个 Ready 信号, 则进入二级安全工作模式, 在这个模式下, 有两个处理机同时处于工作状态, 向调度器输出结果, 如图 8 所示。另外一个处理机处于未启动或启动不成功的状态, 调度器不予理会。在这个模式下工作, 调度器不会进入具有更高等级的一级安全工作模式, 但会由于有处理机达到清洗状态进入三级安全清洗模式。

### 3.4.6 二级安全清洗模式(2-2)

二级安全清洗模式是由一级安全工作模式转换来的, 在这个模式下, 三个处理机的工作状态分别为工作、工作、清洗, 如图 9 所示。处理机清洗完成后, 会恢复一级安全工作模式。如果在这个模式下, 有新的处理机达到了清洗条件, 调度器会转入三级安全清洗模式二进行工作。

### 3.4.7 一级安全工作模式(1)

在系统启动阶段, 如果在超时窗口内成功接收到三个 Ready 信号, 调度器就进入一级安全工作模式。这个模式下, 三个处理机都处于工作状态, 是拟态处理机安全等级最高的状态, 如图 10 所示。

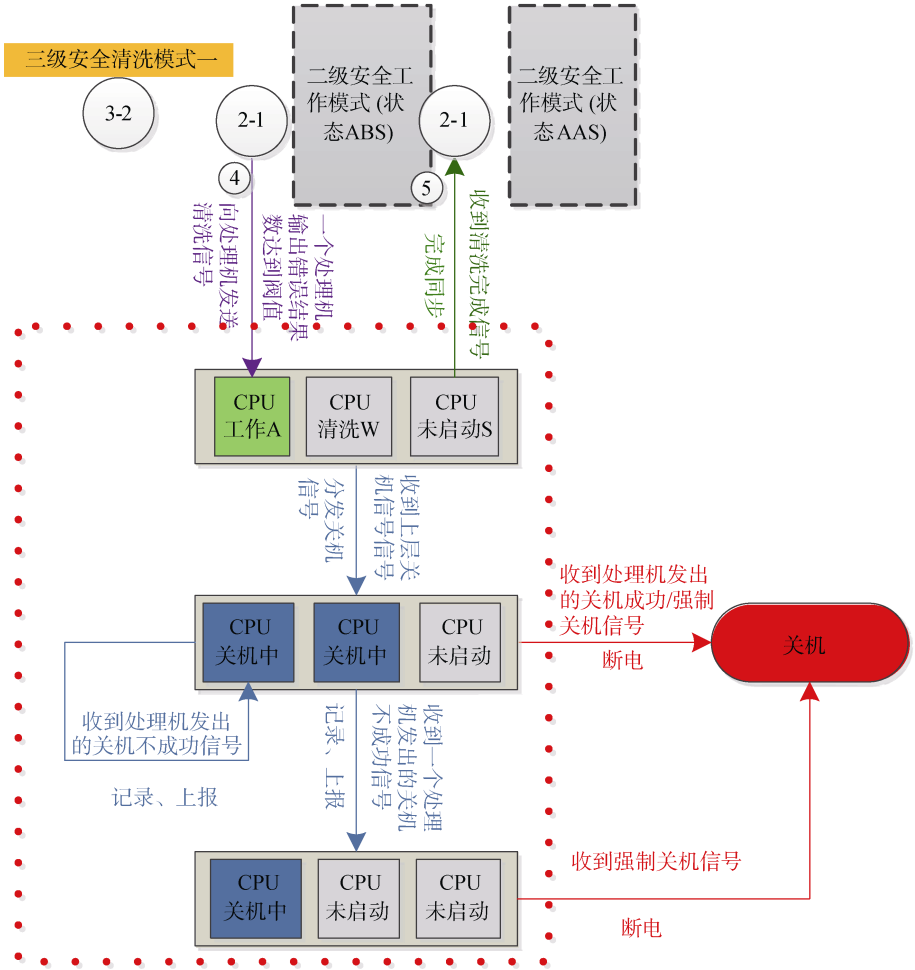



**图 6　三级安全清洗模式一**

### 3.5　基于状态保存的两步清洗方式

当调度器经过仲裁判决认为异构冗余系统中的某个处理机产生异常后，如何处理产生故障的处理机，使系统能够继续正常工作，这就是异构冗余系统的清洗问题。

现有的大部分异构冗余系统采取故障隔离的方法，即 3.4 节中提到的降级策略，当三个处理机都处于正常工作状态时，认为此时系统安全等级为一级安全；当其中一个处理机出现异常，对其进行清洗(即将其隔离不参与后续工作)，此时系统安全等级降为二级安全(此时的判决策略也相应改变)；当系统有两个处理机发生异常，正常工作的只有一个处理机，此时系统安全等级降为三级安全。该清洗策略容易实现，但是会严重降低系统的安全性。

传统的多冗余系统大多是针对可靠性的，某些失效可能是永久性的，难以恢复，且故障恢复实现起来难度较大，所以经常采用故障隔离策略。而拟态安全处理机架构目的是为了保证系统的安全性，攻击通常会利用软硬件后门影响应用程序的正常功能，系统经过重启后通常会消除上次攻击带来的影响，可采用基于重启的方式来实现失效处理机的恢复。

故障恢复常见于如今的数据库和高性能计算系统，因为随着计算性能的提高，系统规模不断增大，系统发生故障的概率也随之增加。如果不采用故障恢复策略，则一旦某个进程或者计算节点发生错误，就会导致整个计算过程失败，系统需要从头开始重新执行，这对那些长时间的计算是不可容忍的。在分布式计算架构中通常采用基于检查点的回卷回复策略。

检查点与回卷恢复(checkpointing and rollback-recovery)技术[22,23]在系统的运行过程中通过某种策略在特定的时间点保存系统的一致状态(检查点)，从而在系统发生故障时，可以通过回卷机制将系统回卷至发生故障前最近的检查点，节省系统从头开始执行所需的大量时间。检查点与回卷恢复技术具有实现和使用简单，占用存储空间小、资源要求低等特点[24-27]，在数据库和高性能计算、分布式处理系统中得到了广泛的应用。

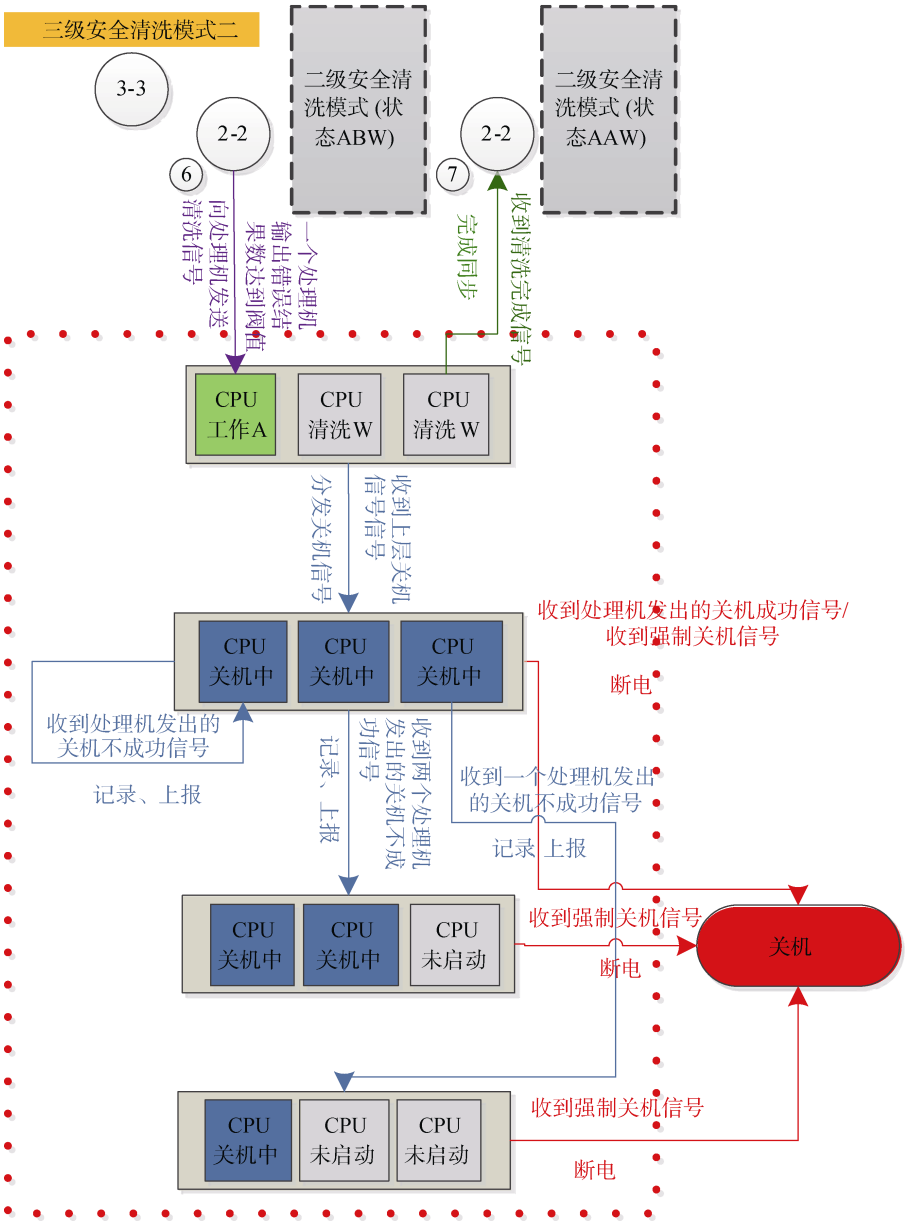



**图 7　三级安全清洗模式二**

在工控领域中, 系统一般都具有严格的实时性及可靠性要求, 某个时刻至少保证有一个处理机在运行。此外, 当某个处理机经过仲裁判决发生故障时, 如果对其进行重启, 需要对所有的历史输入进行保存和重新计算, 时间和存储开销不可承受。可借鉴基于检查点的回卷回复策略, 对处理机的状态进行保存和同步, 使得清洗后的处理机能够快速与其他处理机实现状态同步。在拟态系统中, 由于采用三系统冗余, 所以在保存状态时还需要进行判决比较, 故障恢复时需进行状态同步。

### 3.5.1　程序状态的保存

当某处理机被认定为存在异常时, 拟态调度器会向该处理机发送清洗信号并将其移出比较队列。收到清洗信号的处理机会将处理机中各部件进行复位操作。被复位的处理机若想重新回归正常工作状态需要追赶上其他处理机中程序工作进度。当系统运行很长时间时, 系统上运行的程序积累了大量的历史操作。被复位的处理机若想从头运行程序从而追赶上其他处理机的工作状态显然是不现实的, 因此拟通过保存程序状态来帮助异常处理机复位后尽快恢复。

如图 11 所示, 调度器为每个独立应用保持一组一致性全局状态数据(包括状态字和状态附加数据)。不同组的一致性全局状态数据之间的关联关系由应

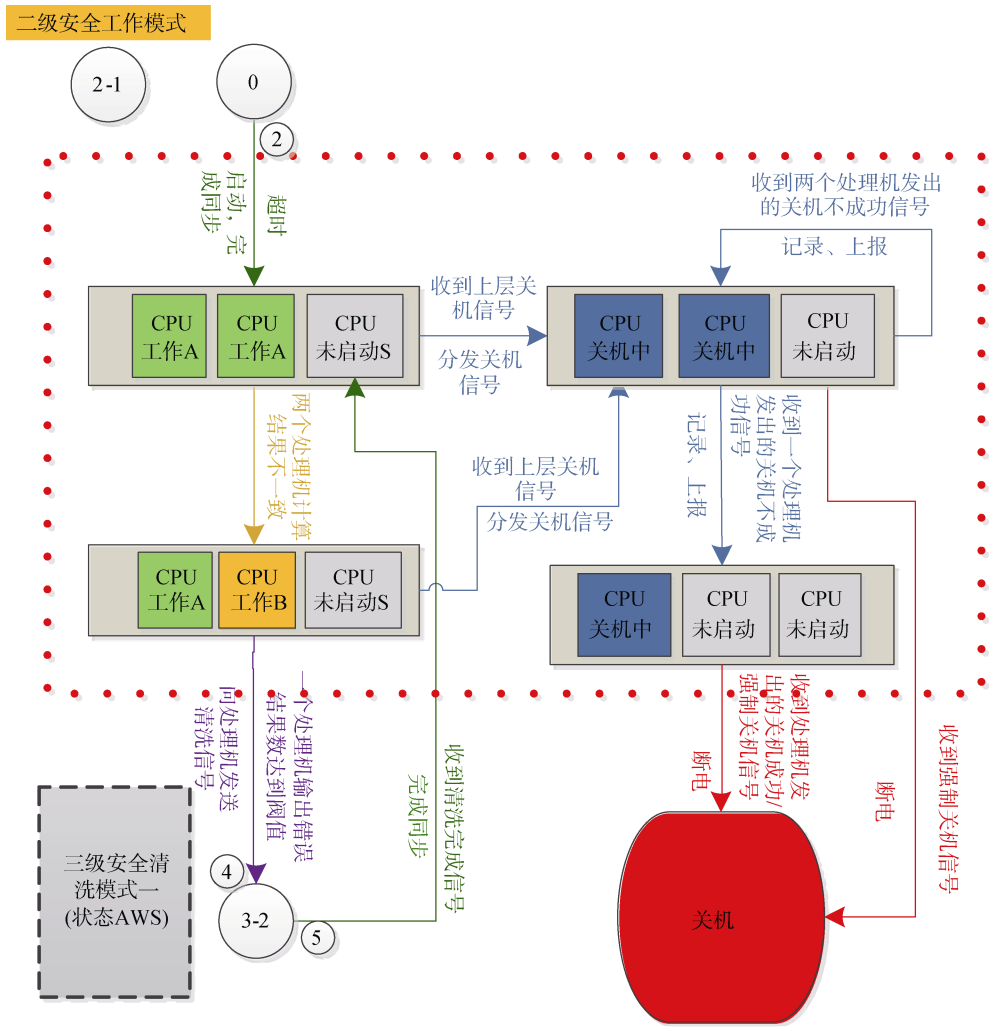



**图 8　二级安全工作模式**

用程序另行处理，调度器认为他们是相互独立的。系统在运行过程中，调度器实时地对各个正常处理机中相同程序的输出结果进行判决并保存在程序的全局状态中，当某个处理机发生异常并重启完成后可以通过保存在调度器中的全局状态数据进行恢复。

为了不同处理机之间实现状态同步，调度器在汇集同一应用在不同处理机的输出结果时，对各个处理机作一次同步，以保证全局数据的一致性。该同步机制的具体操作流程如图 12 所示。

各处理机在对同一应用计算完成后可能会对该应用的状态进行修改，此时各处理机会向调度器发送修改应用全局状态的请求。调度器在预设时间窗口内收到三个处理机的修改全局应用状态请求并对他们的修改状态进行比较，然后将择多判决后的结果保存在应用全局状态数据中。在应用全局状态数据修改完成后，调度器会向各处理机发送全局状态变更通知，各处理机在收到该通知后会统一对全局状态进行一次回读操作，至此应用状态同步完成。

### 3.5.2　两步清洗流程

故障处理机的清洗主要基于全局应用状态数据对异常处理机进行恢复，同时考虑重启处理机的安全性，采用两步清洗流程。其中调度器需维护两组处理机列表，一为比较列表，用以记录系统中正常运行的处理机；二为可用列表，调度器向可用列表中的处理机发送输入并从中接受输出，但并不对采用其输出结果值。当可用列表中的某处理机发送的输出结果值连续正确次数达到某个阈值后，该处理机被认定为行为正常并被移入比较列表。

具体两步清洗流程如图 13 所示，当调度器判定某处理机异常后会向该处理机发送清洗信号，同时将其移出比较队列。收到清洗信号的处理机重启后向调度器发送可用信号。调度器接受到可用信号后将其加入可用列表，第一步清洗完成。调度器将清洗后的处理机加入可用列表后，一旦有全局状态变

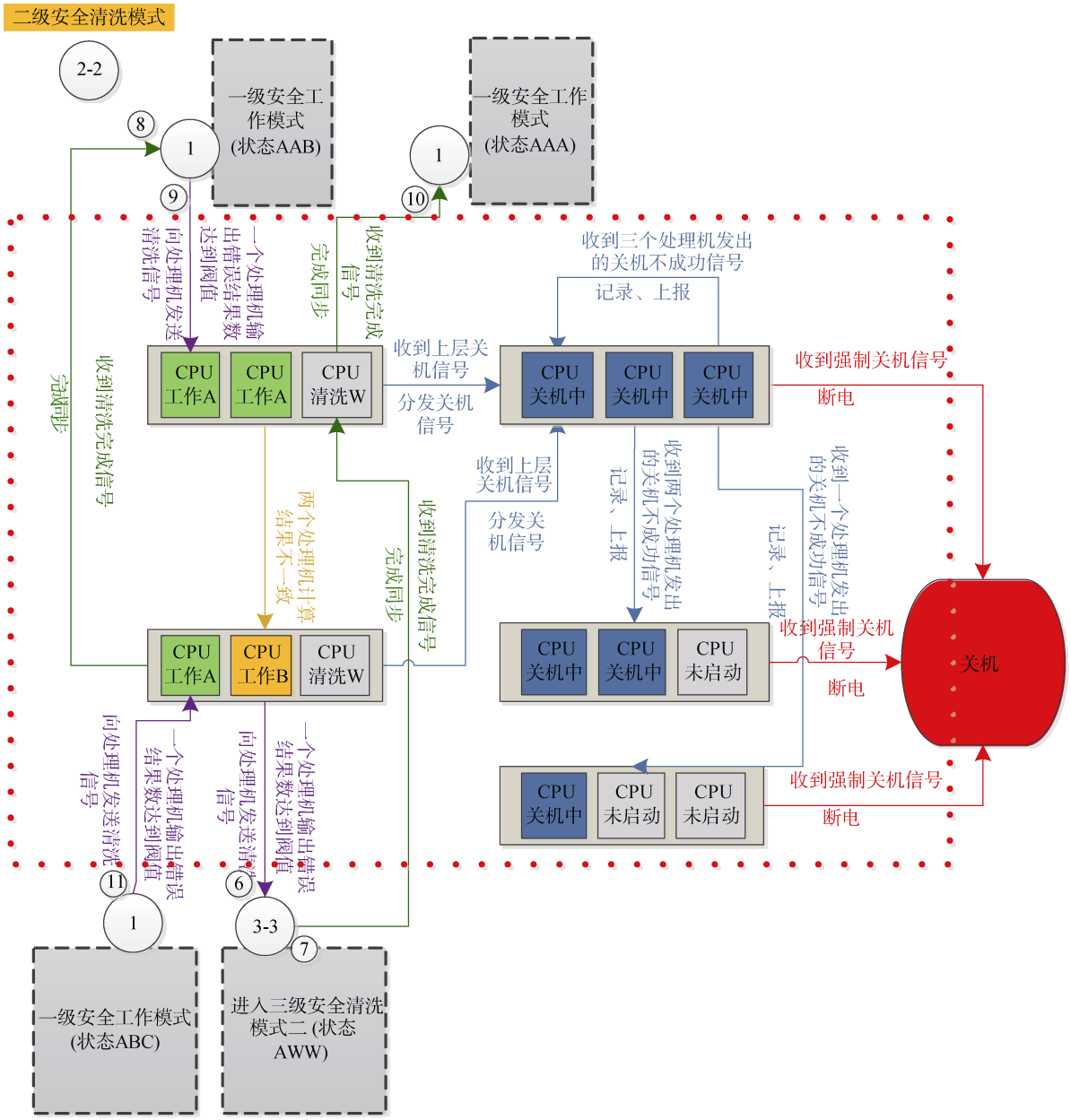



**图 9　二级安全清洗模式**

更, 均会发送给异常处理机用于更新其状态。此外, 调度器会将输入信息发送给处理机用于计算, 当调度器判断该处理机输出的正确次数达到一定阈值后认定该处理机恢复正常工作。至此全部清洗流程完成。

## 4　高可靠调度器判决策略

### 4.1　调度器判决策略设计

受到内部漏洞、后门或外部攻击的影响, 多个异构处理机的输出结果可能出现不一致的情况。对异构处理机的输出结果进行判决包括对结果进行比较、判决和输出, 通过判决达到尽量输出正确结果的目的。为了提高调度器输出正确结果的概率, 需要设计高可靠的判决方法。

在拟态调度器判决策略设计上, 我们参考了在冗余软件系统中进行高可靠控制所使用的表决算法, 包括经典的多数表决算法[28]、n 中取 2 表决算法[29]、一致性表决算法等[30], 并结合了后续改良的自检测多数一致表决算法[31]、自适应一致表决算法[32,33], 最终设计了两种判决策略: 择多判决和单选判决。

择多判决采用三个处理机并行执行任务的工作方式, 对输出结果进行比较, 选择占多数的结果作为正确结果进行输出。单纯的择多判决, 没有考虑各处理机之间的差异性, 当出现多处理机计算结果不一致的情况时, 没有办法鉴别哪个处理机的结果更

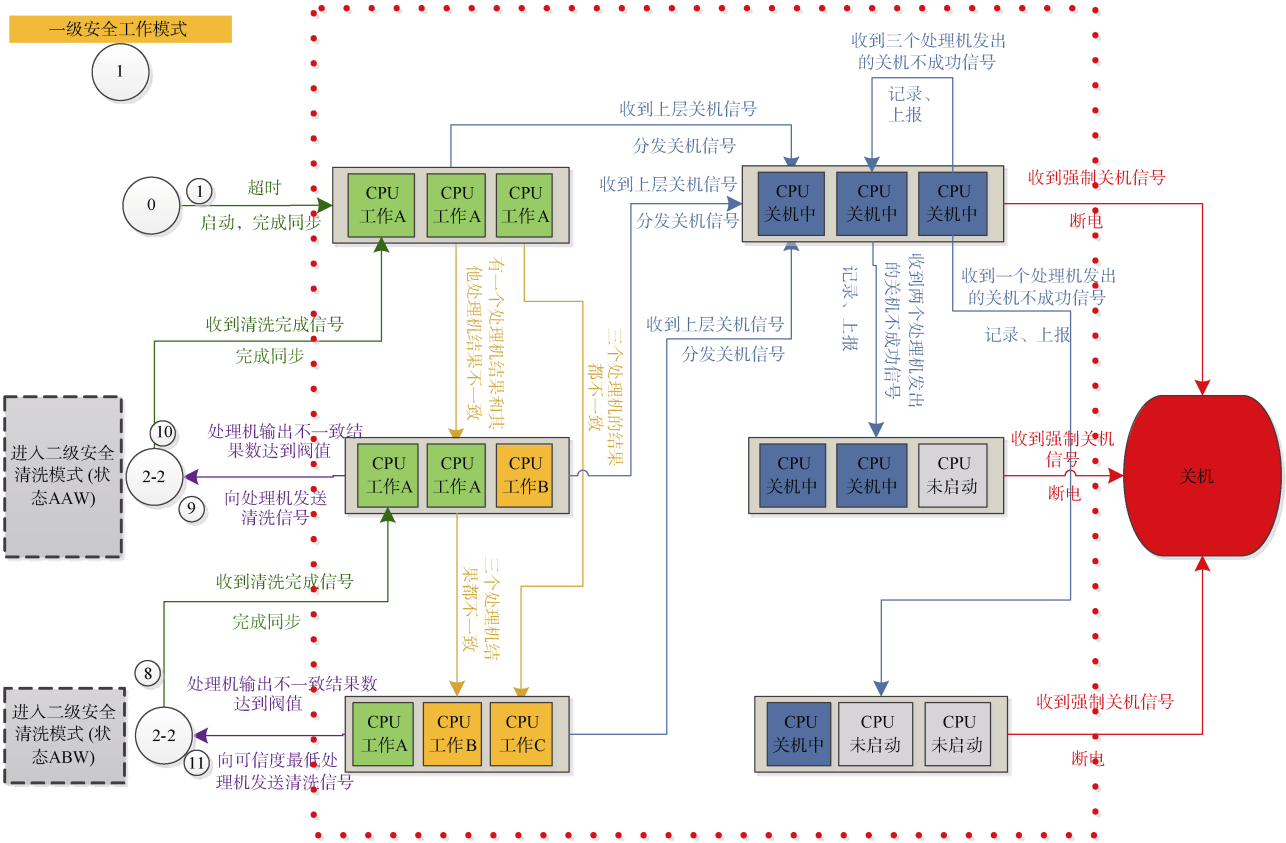



图 10　一级安全工作模式

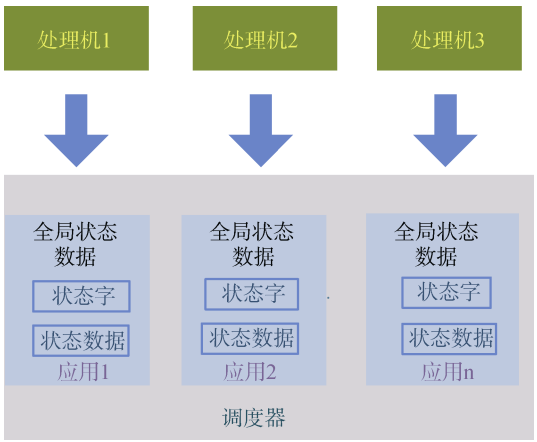



图 11　程序状态保存

可信。因此，在实际应用中，通常会引入一个“可信度”，用来标识处理机之间的差异。在本文中，引入了能反映处理机客观历史表现的经验可信度和能反映系统管理员主观倾向的权值这两种可信度，设计了基于经验可信度的比较择多判决和基于权值的比较择多判决两种策略。

单选判决指调度器每次从各个异构处理机中选择一个处理机作为主处理机，将其结果作为判决结果输出。除了主处理机之外的两个处理机分别称为从处理机与备用处理机，处理机的工作方式为一主一从一备。每隔一段时间，就改变处理机的主从备状态。从安全可靠性来说，单选判决不如择多判决，因此在实际应用中，都会将单选判决与择多判决相结合，形成复合判决策略。在本文中，介绍了一种基于抽样择多的复合单选判决策略。

## 4.2　基于可信度的比较择多判决

采用择多判决时，三个处理机采用同时工作，并持续向调度器发送运算结果的工作模式。调度器比较三个处理机的结果。同时，赋予每个处理机一个可信度，这个可信度用于标识处理机可被信任的程度高低。可信度包括经验可信度和权值。

**择多判决策略**

**情况一：**三个处理机正常工作，三个待判决结果一致。认为三个结果都为正确的，随机选择一个结果进行输出。

**情况二：**三个处理机都正常工作，出现两个相同的待判决结果。认为相同的结果为正确结果，从这两个相同的结果中随机选择一个进行输出。

**情况三：**三个处理机都正常工作，出现三个各不相同的结果。认为可信度最高的处理机的输出结果为正确结果。

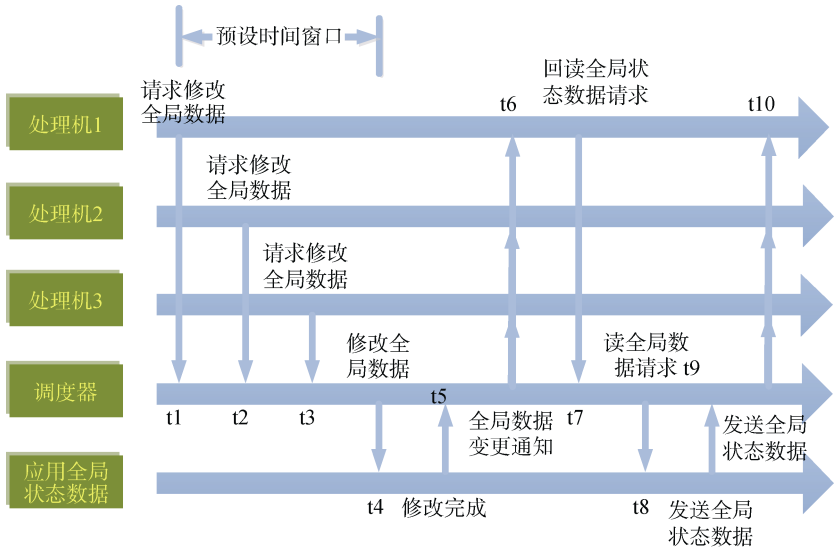



**图 12　全局程序状态保存流程**

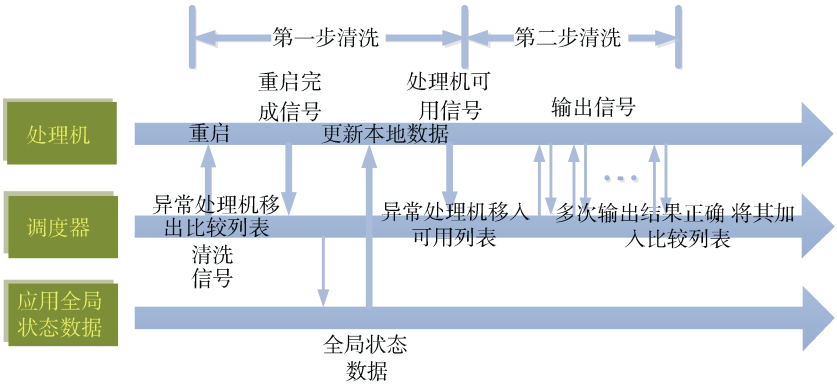



**图 13　两步清洗流程**

**情况四:** 两个处理机正常工作, 另外一个处理机正处于清洗流程。两个正常处理机的输出结果相同, 则认为这两个结果是正确的, 随机选择一个进行输出。

**情况五:** 两个处理机正常工作, 另外一个处理机正处于清洗流程。两个正常处理机的输出结果不相同, 则认为可信度较高的处理机运算正确, 输出其结果为判决结果。

**情况六:** 只有一个处理机正常工作, 另外两个处理机都处于清洗流程, 则选择相信这个唯一的处理机的结果是正确的, 将其结果输出。

**基于经验可信度**

经验可信度的表示: 处理机每输出一次正确的结果(与判决输出一致的结果), 经验可信度就 + 1, 反之则 - 1。

经验可信度描述了该处理机在历次判决中的表现优劣: 值越大, 则表示该处理机输出正确结果的次数越多、被清洗的次数越少; 反之, 则表示该处理机输出正确结果的次数较少、被清洗的次数较多。我们假设具有较高经验可信度的处理机比具有较低经验可信度的处理机更有可能在下一次的输出中提供正确的结果。

如果采用基于经验可信度的择多判决策略, 在系统启动时, 给三个处理机分配相同的经验可信度。如果一个处理机经历了清洗过程, 则经验可信度降低为清洗前的 1/2。将参与过清洗的处理机核的经验可信度显著降低, 相当于给清洗过的处理机核打上了一个“犯过错误”的标记, 对其信任度也显著降低。因为处理机核若是频繁出错, 极有可能是受到了攻击, 即使进行了清洗, 也可能清洗不彻底。特别是针对底层软件及硬件的攻击, 单纯的复位无法清除, 重启过后依然处于受攻击状态。而且, 遭受过一次攻击, 说明攻击者已经掌握了该处理机的基本信息和攻击方法, 打通了对该处理机进行攻击的攻击链, 降低了对该处理机进行再次攻击的成本。因此, 对于清洗过的处理机, 需要降

低其经验可信度。

相比于单纯的择多判决，基于经验可信度的择多判决策略，充分考虑了处理机的历史表现。根据 3.3 节的分析，在三核架构下，判决出错的概率如下表示，其中 $\theta'$ 和 $P_{\text{错误}}$ 成反比关系，$\theta'$ 越高，$P_{\text{错误}}$ 越低。

$$P_{\text{错误}}=C_3^1\times p\times q^2\times\theta+C_3^1\times p\times q^2\times(1-\theta)\times(1-\theta')+C_3^3\times q^3$$

当出现三个处理机运算结果不一致的情况时，拟态调度器根据经验可信度高低选择输出结果。引入经验可信度，可以在一定程度上提高 $\theta'$ (表示当三个待判决结果都不同时，选中正确结果作为输出的概率)的值，进一步降低 $P_{\text{错误}}$ (判决出现错误的概率)的值。

如图 14，取单处理机出错概率 $q\in[0,0.05]$，$\theta'\in[0.2,0.66]$。从图中可以看出，在 $q$ 取值一定时，$\theta'$ 取值越高，$P_{\text{错误}}$ 越低。

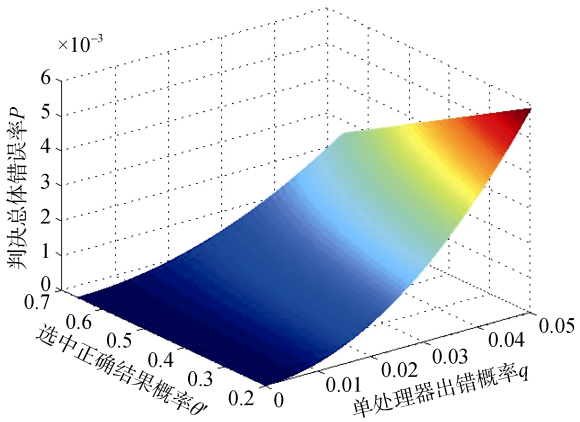



**图 14　$\theta'$ 对 $P_{\text{错误}}$ 的影响**

**基于权值**

处理机的权值反映了在系统中该处理机的重要程度(也即信任度)。权值越高，表明对该处理机的信任度越高。权值和经验可信度的区别在于：经验可信度的值只基于处理机的历史表现，是一个客观值；而权值在初始分配的时候可以反映系统管理员的主观倾向。

如果采用基于权值的择多判决策略，系统启动时，根据管理员预先设定的默认值为处理机分配权值。当处理机清洗完成重新加入计算后，其权值修改为清洗前的 1/2。在判决过程中，权值也会根据处理机的表现实时变化。

基于权重的择多判决策略，采取的策略基础与基于经验可信度的择多判决相似。因此，采用该判决策略，也可以较单个处理机降低一个数量级的错误率。在权重分配合理的前提下，相对于单纯的择多判决策略，该策略也能在一定程度上提高 $\theta'$ (表示当三个待判决结果都不同时，选中正确结果作为输出的概率)的值，使 $P_{\text{错误}}$ (判决出现错误的概率)降低。

### 4.3　基于抽样择多的复合单选判决

采用单选判决时，三个处理机的工作状态分别处于主、从、备。基于抽样择多的复合单选判决是指：以主处理机的输出结果作为判决结果输出，每隔一段时间，将主从处理机的输出结果进行比较，如果二者结果相同，则认为二者状态都正常；如果结果不一致，将备用处理机的结果也加入比较。

在一定情况下，需要改变处理机的主从状态，进行重新洗牌，以此增加系统动态性和随机性。

单选判决根据主从备处理机的输出正误情况，决定处理机切换后的状态。在以下情形需要切换处理机:

1) 主处理机的结果出现错误：主处理机的结果每隔一段时间会和从处理机的结果进行比较，如果比较结果不同，会启用备用处理机，比较三者结果。如果从处理机和备用处理机的结果相同，则表示主处理机的结果出现了错误，需要切换主处理机;

2) 从处理机的结果出现错误：主从处理机结果不同，但主备结果相同时，认为从处理机出现错误，需要切换从处理机;

3) 三个处理机状态都正常，但当前的主从备状态已经持续了一段时间：在三个处理机状态都正常的情况下，依然需要切换处理机。因为系统处于稳定状态的时间越长，被攻击者探测清楚的可能性越大，越容易遭受攻击。

具体的切换和判决措施为:

当主处理机出现错误时，输出从处理机结果。切换时，将从处理机切换为主处理机，将备用处理机切换为从处理机，将主处理机切换为备用处理机。出错的主处理机切换为备用处理机后，调度器向其发出清洗信号，暂不让其参与后续工作，待其清洗完毕后，再让其重新参与后续的工作。

当从处理机出现错误时，输出主处理机的结果作为判决结果。切换时，主处理机不变，将从处理机切换为备用处理机，将备用处理机切换为从处理机。切换后的备用处理机也将被暂时停止服务，进行清洗，待清洗完成后，才参与后续工作。

三个处理机状态都正常的情况下，切换时，可以采取两种策略，一种是随机策略，另一种是轮换策略。

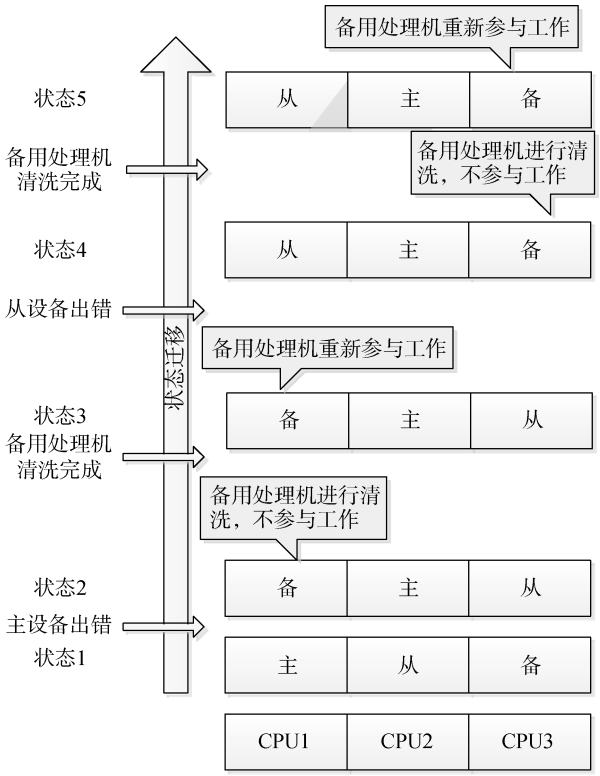



**图 15　单选判决处理机状态切换**

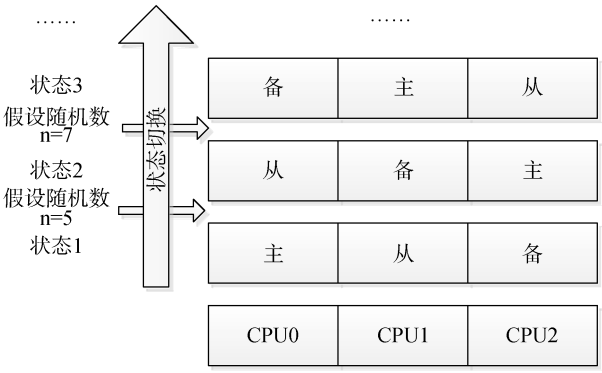



**图 16　处理机状态都正常时进行随机切换**

使用随机策略切换处理机状态时, 将三个处理机分别编号为 0、1、2。生成一个随机数 n, 编号为 (n % 3) 的处理机切换为主处理机, 编号为 ((n%3)+1)%3 的处理机为切换从处理机, 编号为 ((n%3)+2)%3 的处理机切换为备用处理机。n 可以采用程序生成的伪随机数, 也可以是系统时间、中间运算结果、系统实时状态参数等真随机数。

使用轮换策略切换处理机状态时, 切换的时机可以是基于时间的, 也可以是基于程序段的。基于时间轮换的切换策略在 0~T1 时间内采用处理机 1 输出结果, T1~T2 时间内采用处理机 2 输出结果, T2~T3 时间内采用处理机 3 输出结果。基于程序段的轮换策略, 以程序段为单位, 第一个程序段间的输出结果采用处理机 1 的输出结果, 第二个程序段间的输出结果采用处理机 2 的输出结果, 第三个程序段间的输出结果采用处理机 3 的输出结果, 以此类推。一个程序段可能包括一个或多个程序断点。

另外存在一种极小概率事件, 即三个处理机的输出结果都不相同, 则暂不切换处理机状态, 还是输出主处理机的结果, 同时清洗备用处理机。备用处理机清洗完成后, 切换为主处理机, 主处理机切换为从处理机, 从处理机切换为备用处理机。切换完成后, 清洗备用处理机。清洗完成后, 再进行一轮切换, 再次清洗备用处理机。最后一次清洗完成后, 再切换一次。这种极端情况下, 为了保险起见, 依次清洗了三个处理机, 最大程度保证了判决结果的正确性。

在判决时, 正常情况下, 调度器输出主处理机的结果; 如果主处理机出错, 从处理机和备用处理机的结果正确, 则输出从处理机的结果, 并切换处理机状态; 如果从处理机出错, 主处理机和备用处理机的结果正确, 则输出主处理机的结果, 并切换处理机状态。

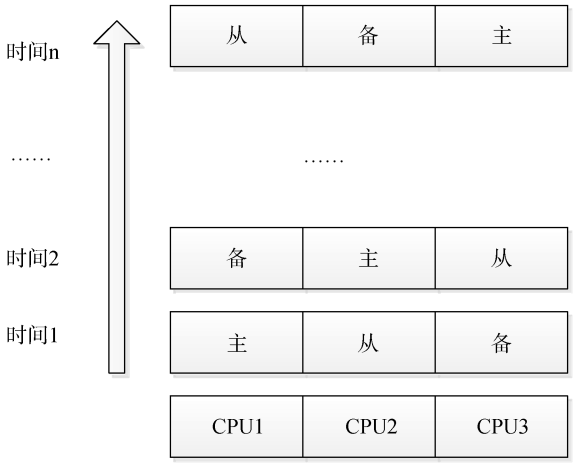



**图 17　基于时间的轮换策略**

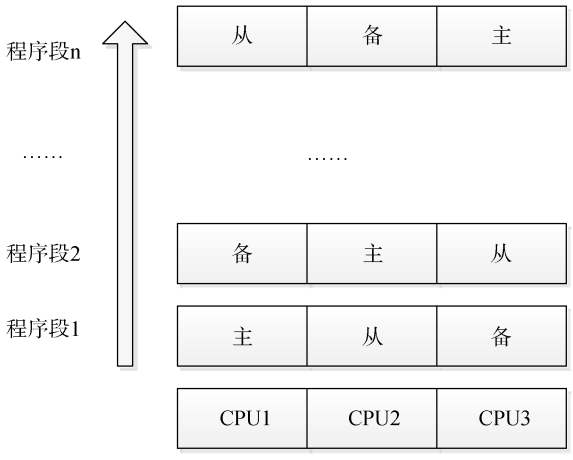



**图 18　基于程序段的轮换策略**

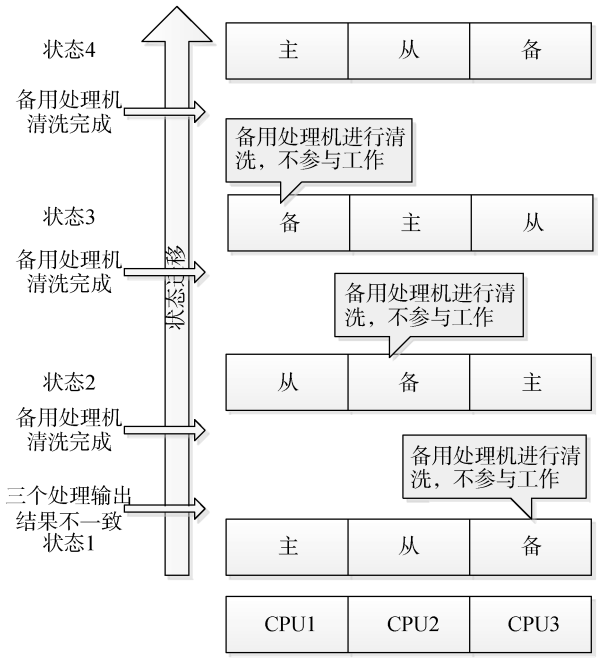



**图 19　三个处理机输出不一致**

由于处理机出错引起的切换，切换完成后，先要清洗备用处理机。如果备用处理机还没有清洗完成，又出现了主从处理机结果不一致的情况，先输出主处理机的结果，待备用处理机清洗完成后，再将备用处理机的结果加入比较，找出错误处理机，切换其状态并清洗。

**单选判决策略有效性分析**

单选判决与择多判决的主要区别在于，单选判决在大部分时候信任主处理机的输出，隔一段时间才会检查一下主处理机的输出是否正确。如果在此期间主处理机出现故障，是没有办法判别出来的。但是，处理机切换的周期理论上是小于完成一个攻击的全周期的。动态地切换处理机能降低单个处理机出错的概率。另外，单选判决在比较的时候其实采取的也是“多数行为准则”，同择多判决一样，也能判别出异常处理机，并进行及时的清洗恢复处理，同时利用正确的处理机对外正常提供服务。从可靠性、快速反应和正确率上，都较单个处理机有显著提高。

## 5　仿真验证与分析

本文提出的为面向通用工控系统的处理机架构，仿真验证环节中的网络拓扑保留最基本的网络节点模型，即输入输出单元(人机交互单元)、仲裁器(控制单元)和处理单元，过程监控网与现场控制网使用 TCP/IP 网络协议实现通信与数据传输。其中输入输出单元、仲裁器与处理机运行环境均为目前主流操作系统，为模拟拟态安全处理机中处理机的异构特征，本课题中的三个处理机采用互不相同的操作系统。

为了验证面向工控领域的通用拟态安全处理机的有效性，初步采用软件方式进行调度器的模拟仿真，仿真环境的硬件架构如图 20 所示，三个处理机、模拟输入输出主机和仲裁器之间通过以太网进行连接，其中模拟输入输出主机进行数据的模拟输入，数据统一到达仲裁器之后被分发至三个处理机，处理机计算完成后发给仲裁器进行仲裁判决后再传至模拟输入输出主机。内部机制示意图如图 21 所示，在仲裁器内部，运行一个 Server 代理、三个 Client 代理；分别和模拟输入输出主机和三个处理机进行通信交互；整个仲裁器由 C 语言编写，实现上文中介绍的调度、判决与清洗策略。三个处理机操作系统分别为 Windows 7 系统、中标麒麟系统与 Ubuntu 12.04 系统。

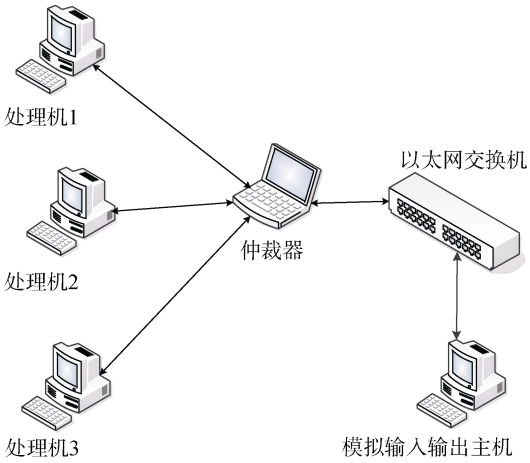



**图 20　仿真环境硬件结构**

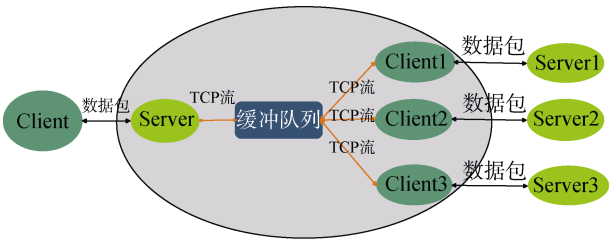



**图 21　仿真机制结构**

工业控制系统网络不同于普通的 IT 网络，它是一个半封闭的网络，具有实时性通信特点，各节点高度同步；通信协议多样，包括 TCP/IP 协议、Modbus 协议、CAN 总线协议等。随着工业物联网的推进，少数工控系统实现了与互联网的互通互联，但出于安全性的考虑，大多数工控系统与互联网物理隔离。仿真验证环节主要针对本课题提出的拟态安全处理机架构的安全性能，而安全性能依赖于拟

态系统中仲裁器的控制。由于实际工控系统数据的限制, 本仿真测试假定 PC 机能够模拟工控系统的处理运行性能, 而现实中工控系统使用的操作系统多为 Windows、Linux 系统, 且仲裁程序的软实现完全能够模拟实际工控系统中的仲裁控制单元。因此, 针对工控系统的安全性能, 本仿真验证具有一定可行性。

针对处理机的攻击与处理机的静态性、确定性、相似性和持续性的具有强依赖关系, 某种攻击方法一般只适用与特定的处理器、操作系统和中间件, 任何一个环节的变化都会导致攻击的失败。拟态安全处理机采用异构冗余设计, 这些异构冗余部件之间基本上不具有相同的漏洞或后门, 对其中一个异构冗余部件生效的攻击在其他部件上成功的概率极低, 而不能同时造成两个(或以上)异构冗余部件产生相同错误的攻击对于拟态安全系统来说都是可以识别和化解的。拟态安全测试主要是分析工控处理机可能面临的安全风险, 模拟各种攻击场景, 以实验方式验证拟态处理机系统的安全性和有效性。

## 5.1 面向工控领域攻击模型的测评场景和方案

针对第三章提出的针对工控领域的攻击模型, 本文中安全测试验证拟以系统探测与弱点扫描、系统突破(漏洞利用)以及后门攻击为测试评估场景及手段, 与目前的典型同等级工控系统为参照, 评估拟态安全处理机的安全性。

测试对象:

拟态安全测试针对拟态系统中的处理单元, 即三个处理机, 测试中假定仲裁控制单元是安全可信的。同时, 拟态系统对外界透明, 且呈现为一个独立的处理器系统。

测试结果评估方法:

对于拟态系统安全性能的评估通过与目前典型的同等级工控系统作对比。在本测试场景中采用单处理机构成的模拟工控系统作为比较对象。

测试方案:

拟采用的具体测试方案如下:

**(1) 系统探测**

系统探测技术通常包括主机探测、端口探测、操作系统探测、弱点探测、协议检测等, 能够有效扫描出目标网络中存活主机、交换机和路由器等关键设备, 然而由于工控系统网络中通信协议多样, 多数探测技术未必适合工控网络, 但通过协议转换使得扫描工具变得可行, 然后通过数据包的解析可提取并鉴别网络中的工控设备信息[34], 同时, 通过端口、协议等信息快速定位工控网络中的 SCADA 系统。例如若扫描出某设备的 502 端口使用 Modbus 协议, 即可推断与该备连接的可能是 HMI 系统。

在本仿真场景中, 模拟工控系统调试接口设置在外围网络, 即拟态边界之外。通过使用外部调试机连接调试接口对工控网络进行扫描探测, 具体过程为:

Step1. 通过调试接口连接调试机, 在调试机上通过 Nmap 扫描工具对内部网络进行探测, 由于防火墙或 IDS 的作用, PING 命令通常不可用, 我们使用 -PN 命令以非 PING 的方式进行半开放式扫描, 获得整个内部网络主机存活情况;

Step2. 通过 OS 检测命令分析目标主机的操作系统类型与版本信息, 接下来对目标主机进行端口扫描、操作系统探测、服务探测、IP 协议探测等, 分析目标主机开放端口、服务及相关应用类型等;

通过以上步骤完成对模拟工控系统探测扫描测试, 扫描结果如表 3a 所示。

**(2) 系统突破**

系统突破是指攻击者利用非授权接入、漏洞等非法获取系统内主机权限, 从而给系统安全造成威胁。本场景下的测试方案假定系统探测部分的攻击已经完成, 并进一步对系统漏洞进行利用实现代码注入攻击。

操作系统中的漏洞是常见的, 根据漏洞成因通常可以将它们分为程序设计 bug、系统配置不当、弱口令以及初始设计不当等造成的漏洞。本实验场景中, 我们假定系统口令、配置安全, 利用普遍存在的程序 BUG 与设计漏洞, 分别通过 Windows 7 系统动态链接库进行缓冲区溢出攻击与利用 Linux 内核代码漏洞进行本地提权, 并使用系统权限对系统强制关机的方式模拟攻击的发生。

Step1. 针对 Windows7 操作系统处理机, 我们构造一种系统栈溢出攻击方式, 通过利用其系统程序默认加载的动态链接库 user32.dll 中的 jmp esp 指令作为跳板定位 shellcode;

Step2. 针对 Linux 系统, 我们使用的 kernel 版本为 3.16(2015 年 6 月前版本), 通过本地提权漏洞 CVE-2015-1328 及其 POC 代码获取 ROOT 权限;

Step3. 在调试机布置 Windows 系统 shellcode 代码与 Linux 系统本地提权代码同时加入系统关机指令, 并通过调试接口将调试机与工控网络进行连接, 利用调试机分别向三个处理机发送执行代码, 测试获取系统 shell 与执行关闭系统指令的实现;

通过以上过程进一步测试本系统抵抗系统突破与攻击的能力。

**(3) 后门攻击**

工控系统的数据处理通常包括许多敏感信息, 这些信息包括地理坐标、温度信息、运行轨迹等, 针对工控系统的后门程序可以利用这些敏感信息实现快速、精确定位与攻击。在本仿真实验情景中, 我们通过白盒测试的方法, 即假定一个处理机(例如 Ubuntu 系统)中存在后门程序进行测试, 具体过程为:

假定 Ubuntu 系统中存在 GPS 后门程序, 我们通过简单的代码实现一个模拟 GPS 坐标触发后门攻击的程序, 程序实现为当输入的坐标数据为预先设置的值时即达到触发条件, 其攻击触发结果为导致系统关机; 通过输入输出主机发送预设 GPS 坐标数据, 测试系统抵御后门攻击能力。

## 5.2　测评结果与分析

测试结果列在表 3 中。为直观显示测试结果, 我们将应用于拟态架构的测试结果与单处理机架构(存在调试接口, 不存在仲裁器模块, 测试方法相同)进行对比(表 4 所示, 其中通过项表示系统可以抵御该项威胁)。

**表 3　测试结果**

| 测试大项 | 测试子项 | 测试结果 |
|---|---|---|
| 系统探测 | 非 PING 主机发现 | 256 IP address <2 hosts up> scanned |
| | 端口扫描 | All 1000 ports scanned are filtered |
| | OS 检测 | Linux、Windows |
| | 服务探测 | ICMP、IPSec |
| | IP 协议扫描 | icmp: open<br>igmp: closed<br>rsvp: closed<br>gre: closed<br>ospfigp: closed |
| 系统突破 | Windows 7 系统利用 user32.dll 栈溢出攻击 | Windows 7 系统关机, 其他系统正常运行, 拟态系统安全等级降低 |
| | Linux 系统 CVE-2015-1328 漏洞本地提权 | Ubuntu 12.04 系统关机, 其他系统正常运行, 拟态系统安全等级降低 |
| 后门攻击 | 自定义 GPS 后门触发 | 存在后门系统(Ubuntu)关机, 其他系统正常运行, 拟态系统安全等级降低 |

**表 4　测试结果(拟态架构与单处理机结构对比)**

| 编号 | 测试大项 | 测试项数量 | 拟态系统通过项数 | 单处理机系统测试数量 | 单处理机系统通过项数 |
|---|---|---|---|---|---|
| 1 | 系统探测 | 5 | 5 | 5 | 0 |
| 2 | 系统突破 | 2 | 2 | 1 | 0 |
| 3 | 后门攻击 | 1 | 1 | 1 | 0 |

在本文提出的拟态架构中, 拟态边界设置在系统 I/O 接口, 即拟态防护范围为异构处理机, 包括运行在处理机上的操作系统等, 而仲裁器的存在对于拟态边界外部是透明的。在这种结构中, 任何的输入输出数据都要经过仲裁器, 任何的输出都将经过仲裁器的一致性判决。实验结果表明:

(1) 系统探测阶段, 使用 Nmap 高级扫描方式可以绕过系统防火墙与 IDS 机制, 可以实现对于目标网络存活主机、开放端口、使用协议、系统类型等的检测。表 3 列出了基于 Nmap 扫描工具通过调试接口对系统进行探测的测试结果。以 OS 探测为例, Nmap 对图 20 中的网络进行扫描, 扫描到输入输出主机为 windows 系统, 扫描拟态处理机时, 仲裁器将扫描命令传递给三个异构处理机, 异构处理机输出不同内容, 仲裁器判断后以可信度比较高的处理机(Linux 处理机)输出为最终输出, 所以对外呈现为 Linux 主机; 其他扫描由于两个/三个异构处理机都未开放相关协议, 经过仲裁器裁决后对外呈现关闭状态。测试表明拟态系统使得探测者只能得到拟态处理机对外呈现的信息, 无法探测系统全貌。

(2) 系统突破阶段, 在假定外围网络被突破, 即仲裁器被利用的场景下, 我们首先针对 windows 系统利用其系统缓冲区溢出漏洞通过调试机向拟态系统发起注入攻击, 造成 windows 系统处理机异常, 但其他系统处理机由于没有一致性漏洞而不受影响; Linux 系统本地提权漏洞攻击测试结果一致。需要说明的是, 在本项测试中我们选择比较典型的缓冲区溢出漏洞与通用版本漏洞 CVE 为代表, 选择系统关闭指令作为攻击指令以简化实验, 同时取得较为明显的实验效果。

在针对系统突破攻击场景中, 攻击者所利用的系统漏洞是不受限制的, 即漏洞包含但不局限于常

见的缓冲区溢出漏洞, 以及系统内核级漏洞, 甚至 0-day 漏洞等未知漏洞; 攻击者通过系统漏洞所执行的攻击行为是不受限制的, 即攻击行为不局限于系统提权、系统关闭或系统崩溃, 包含能够破坏系统正常功能输出的攻击威胁。测试结果表明, 针对系统漏洞进行的攻击行为能够对相应子处理机系统造成威胁, 然而在拟态系统中, 这种攻击结果并不影响系统正常功能输出, 由于处理机的异构性, 系统漏洞攻击成功的条件是攻击者所利用的漏洞为所有异构子处理机同时存在的一致性漏洞, 这种情况出现的概率极小。所以说, 拟态架构能够屏蔽由非一致性操作系统漏洞造成的威胁。

(3) 后门攻击阶段, 实验测试利用常用的系统数据 GPS 数据作为后门触发条件, 以系统关闭作为后门触发造成的攻击行为, 模拟工控系统遭受后门威胁的场景。测试结果表明利用应用程序后门能够造成存在后门的系统异常而导致拟态系统处理机进入低安全等级工作模式, 但单个子处理机的异常或崩溃不影响整个系统功能输出, 拟态处理机系统仍能正常运行, 该试验表明拟态系统针对后门威胁较高的安全性能。

### 5.3 总结

本仿真验证部分分析了面向工控领域的网络攻击与面向传统互联网的网络攻击的区别与联系, 并以根据面向工控领域的攻击模型为场景, 测试评估本课题提出的拟态处理机系统应对攻击威胁的能力, 测评结果表明与典型的单处理机系统相比, 拟态架构系统中的仲裁控制、一致性判决过程与冗余机制可以大大降低系统遭受攻击以及因攻击而无法正常运行的可能。下一步的研究工作是将本系统通过 FPGA 实现, 以进一步验证本课题中拟态系统的有效性与安全性。

## 6 总结和展望

本文针对工控系统的通用处理模块设计了一种拟态安全架构, 利用三余度异构冗余处理机作为设计基础, 采用择多判决或者主从备的工作模式, 降低了处理机系统中处理机、操作系统、中间件中后门/漏洞的利用风险。

在设计过程中, 针对工控系统对可靠性要求较高的特性, 需要对各种异常情况进行考虑, 并给出相应的应对策略, 比如某个处理机因为异常不能启动的情况, 或者同时出现两个处理机需要清洗的情况等。针对实时性要求, 在仲裁判决和应用程序状态保存时设定了截至时间, 如果在截至时间内还未收到相应的反馈, 则表明该处理机出现异常, 并进入相应的异常处理流程。

工控系统是一个复杂的系统, 具有多种不同模块。大型工业控制系统更加复杂, 分为企业管理层、过程监控层、现场执行层等多个层次, 每个层次的功能和特点均不相同, 如何在满足工控要求的情况下进行拟态安全改造, 仍是一个有待解决的问题。此外, 在实时系统中, 系统可能对不同的激励反应时间要求不一, 如何合理地满足系统的实时性要求也是一个需要深入研究的问题。

## 参考文献

[1] “2015 工业控制网络安全态势报告—匡恩,” http://www.kuangn.com/2016/04/0411, 2016.

[2] “2014 绿盟科技工控系统安全态势报告,” http://www.nsfocus.com.cn/content/details_62_887.html, 2014.

[3] 杨伟. “容错飞行控制系统,” 西北工业大学出版, 2007.

[4] 潘计辉, 张盛兵, 张小林等. 三余度机载计算机设计与实现[博士学位论文]. 西北工业大学学报, 2013, 31(5): pp. 798-802.

[5] 刘佳. 小型化飞行控制计算机设计与实现[博士学位论文]. 南京航空航天大学, 2008.

[6] 杨林芳. 无人机容错飞行控制系统研[博士学位论文]. 南京航空航天大学, 2007.

[7] 关玲玲. 非相似交叉监控双余度飞控计算机架构设计与可靠性评估[博士学位论文]. 兰州大学,2010.

[8] 周小超, 陆熊. “非相似余度计算机系统及其可靠析”, 计算机与现代化, 2013(5), pp. 135-137.

[9] 韩炜. 非相似容错计算机系统设计及其验证技术研究[博士学位论文]. 西北工业大学, 2002.

[10] 秦旭东, 陈宗基, 李卫琪. “大型民机的非相似余度飞控计算机研究”. 航空学报. 2008, 29(3), pp. 686-694.

[11] Drozeski GR. “A fault-tolerant control architecture for unmanned aerial vehicles[Dissertation].” Georgia Institute of Technology, 2005.

[12] Chen ZJ, Qin XD, Gao JY. “Dissimilar Redundant Flight Control Computer System,” ACTA AERONAUTICA ET ASTRONAUTICA SINICA. 2005,26(3),pp. 320-327.

[13] Gururajan S, Adviser-Napolitano M. “Design and simulation of advanced fault tolerant flight control schemes,” West Virginia University, 2006.

[14] 石贤良. 飞行控制计算机系统余度管理技术研究[博士学位论文]. 西北工业大学, 2006.

[15] 刘志颖, 郑松, “异构三重冗余控制系统的设计与可靠性评估.” 电气技术, 2014, (4), pp. 54-59.

[16] 满梦华, 原亮, 丁国良等. “多核异构冗余模型设计与可靠性分析.” 军械工程学院学报, 2010, 22(1), pp. 67-71.

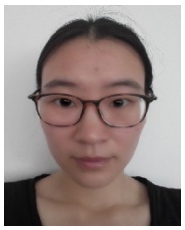
[17] Jajodia, S.,Ghosh, A. K.,Swarup, V., etc. “Moving Target Defense—Creating Asymmetric Uncertainty for Cyber Threats.” In Springer Press, 2011.

[18] Evans, D.,Nguyen-Tuong, A.,Knight, J. “Effectiveness of moving target defenses. In Moving Target Defense,” Springer, 2011, pp. 29-48.

[19] 百度百科. 拟态计算机. 2013.
[20] Robin. “China successfully developed mimicry computing performance by hundreds of times,” 2013.
[21] Bass J M, Latif-Shabgahi G, Bennett S. “Experimental Comparison of Voting Algorithms in Cases of Disagreement,” Proc 23rd EUROMICRO Conference on New Frontiers of Information Technology. Washington: IEEE Computer Society, 1997, pp. 516-523.
[22] Bidyut Gupta, Shahram Rahimi, Yixin Yang. “A Novel Roll-Back Mechanism for Performance Enhancement of Asynchronous Checkpointing and Recovery,” Informatica (Slovenia), 2007, 31(1), pp. 1-13.
[23] Nishanth Chandrasekaran, Suman Kalyan Mandal. “A Two level Cache Based Checkpointing and Rollback Recovery Scheme using MultIPle Epochs.” TR-Texas A&M University, 2006.
[24] George Bosilca, Remi Delmas, Jack Dongarra, Julien Langou. “Algorithm-based fault tolerance applied to high performance computing.” J. Parallel Distrib. Comput., 2009, 69(4), pp. 410-416.
[25] Sunil Kumar Gupta, R. K Chauhan, Parveen Kumar. “Backward Error Recovery Protocols in Distributed Mobile Systems: A Survey.” Journal of Theoretical and Applied Information Technology, 2008, 30(4), pp. 225-240.
[26] Claudia Rusu, Cristian Grecu, Lorena Anghel. “Blocking and Non-blocking Checkpointing and Rollback Recovery for Networks-on-Chip.” Dependable and Secure Nanocomputing, 2008.
[27] D. Manivannan. “Checkpointing and Rollback Recovery in Distributed Systems: Existing Solutions, Open Issues and Proposed Solutions.” In Proceedings of the 12th WSEAS International Conference on Systems, Heraklion, Crete Island, Greece, 2008, pp. 22-24.
[28] Lorczak P R, Caglayan A K, Eckhardt D E. “A Theoretical Investigation of Generalised Voters,” in IEEE 19th Int. Ann. Symp. on Fault-Tolerant Computing Systems. Chicago, June 1989, pp. 444-451.
[29] Kanekawa K, Maejima H, Kato H, et al. “Dependable On-board Computer Systems with a New Method: Stepwise Negotiated Voting.” in IEEE 19th Int. Ann. Symp. on Fault-Tolerant Computing Systems. Chicago, June 1989, pp. 13-19.
[30] Bass J M. “Voting in Real-Time Distributed Computer Control Systems.” Sheffield, UK: Department of Automatic Control and System Engineering, The University of Sheffield.
[31] 周海涛, 朱纪洪, “基于自检测的多数一致表决算法.” 清华大学学报(自然科学版), 2005, 45(4), pp. 488-491.
[32] 俞功兵, 王俊峰, “基于自检测的自适应一致表决算法.” 电子设计工程, 2012, 20(21), pp. 29-31.
[33] 欧阳城添, 王曦, 郑剑, “自适应一致表决算法.” 计算机科学, 2011, 38(7), pp. 130-133.
[34] 王玉敏, “工业控制系统的常见攻击。”中国仪器仪表, 2012(3), pp.60-65.
[35] 王欢欢, 张冬梅, 于亮, “一种针对工控系统的网络探测方法.” 全国青年通信学术年会, 2014.

**魏帅**　于 2012 年在信息工程大学并行编译与高性能计算专业获得博士学位。现任国家数字交换系统工程中心讲师。研究领域为嵌入式计算。研究兴趣包括: 人工智能。Email: weis0906@163.com

**顾泽宇**　于2015年在信息工程大学信息安全专业获得学士学位。现于国家数字交换系统工程中心攻读研究生。研究领域为网络安全。研究兴趣包括: 主动防御。Email: starrynight9f@163.com



**于洪**　于 2013 年在国防科学技术大学网络安全专业获得硕士学位。现任国家数字交换系统工程中心助理研究员。研究领域为嵌入式系统。研究兴趣包括: 计算机架构安全等。Email: yuhong_3210@163.com

**张兴明**　现任国家数字交换系统工程中心教授, 研究领域为交换及拟态技术。研究兴趣包括: 拟态计算及拟态安全。Email: ndsczxm@sina.com