适应动态安全需求的实时任务调度算法研究

夏 平，周兴社，谢必昌

(西北工业大学 计算机学院，陕西 西安 710072)

摘 要：现有的实时容错调度算法没有将实时任务的动态安全需求与其可调度性结合起来考虑。针对这一问题，文章展开以下研究工作：基于安全分级思想，构建了一个适应动态安全需求的实时调度模型，该模型详细地描述了实时任务、系统安全服务、任务容错等实时调度过程涉及的关键要素。以此模型为基础，提出了一种自适应实时容错调度算法(AFTS)，该算法支持优先级抢占式调度策略，以牺牲普通任务的运行为代价来保证关键任务的可调度性，并采用主副本备份技术实现了关键任务的容错功能。当系统安全级别被动态调整时，该算法能够为实时任务选择满足当前安全需求的最优安全策略。仿真实验表明，文中提出的算法与同类算法相比，在系统动态安全需求的适应性，以及关键任务的可调度性和容错能力等方面有较好的表现。

关 键 词：算法，计算机仿真，设计，效率，数学模型，实时系统，调度，数据安全，自适应容错调度算法，动态安全需求

中图分类号：TP316 **文献标识码**：A **文章编号**：1000-2758(2012)05-0657-06

随着应用环境的日益复杂，实时系统面临着越来越多的威胁和挑战，特别来自安全方面的风险将导致系统发生失泄密事故，甚至因黑客攻击而运行崩溃。人们开始对实时系统的安全性越来越重视，文献[1]将安全性列为系统可信性的重要特征属性之一。目前，许多实时系统已经在底层提供了各种类型的安全服务，这些安全服务封装并实现了不同的安全保护功能，每个实时任务通过组合调用这些安全服务以满足自身的安全性需求。由于实时任务在调用安全服务时会占用大量的计算、存储资源，由此产生的安全开销会对系统的可调度性造成严重的影响，致使系统无法实现正常的调度，因此实时调度算法必须将这部分安全开销纳入考虑范围。同时，为了适应工作环境的变化，实时系统在运行过程中会动态改变实时任务的安全需求，实时系统对安全级别的调整必然会导致实时任务的安全需求发生相应改变。此时，实时任务如何动态选择最优的安全策略，使得在满足安全性需求的同时，兼顾满足实时性和可靠性需求，将成为研究新型实时调度算法必须考虑的问题。

目前关于实时任务调度技术的研究已经非常成熟，这方面已经获得了很多研究成果。文献[2-4]提出了多种动态实时调度算法，每种算法都能够有效提高调度的效率，但是这些算法均未考虑实时任务的安全性需求对调度过程的影响，因而不适用于安全关键(Security-Critical)实时应用。文献[5]提出了一种用于异构集群实时系统的双阶段调度策略，该策略考虑了软实时任务的安全需求和时间限制，并给出了自适应调度算法 DSRF 和安全级别均衡算法 FMSL，能够按照系统负载来调节任务的调度成功率与安全级别之间的平衡。文献[6]针对安全关键实时应用提出了一种安全感知的启发策略 SAREC，该策略将实时任务的安全性需求与调度结合起来考虑。通过将 EDF 调度策略融入 SAREC，形成一种安全感知的实时调度算法 SAEDF，该算法能够在保证实时任务可调度性的同时，尽量提高任务的安全级别。以上 2 种算法仅适用于软实时应用，且不支持抢占式调度和未考虑实时任务的容错功能。文献[7,8]提出了 2 种启发式可信调度算法，均采用主副本技术来实现周期实时任务的容错功

收稿日期：2011-11-10 **基金项目**：国家自然科学基金(60736017)资助

作者简介：夏 平(1982—)，西北工业大学博士研究生，主要从事高性能计算及实时计算的研究。

能,并在调度过程中考虑了安全开销的影响。但这两类算法属于静态调度算法,无法满足系统动态变化的安全需求。

针对以上算法存在的不足,我们提出一种适应动态安全需求的新型实时容错调度算法 AFTS(Adaptive Fault-Tolerant Scheduling Algorithm) ,该算法依照系统安全级别来为每个实时任务动态设置最优的安全策略,并以牺牲普通任务可调度性为代价来保证关键任务的可调度性,通过采用主副本备份技术实现了关键任务的容错功能。仿真实验表明,本文提出的算法与同类算法相比,在系统动态安全的适应性、关键任务的可调度性和可靠性等方面有较好的表现。

1 系统调度模型

1.1 问题描述及假设

本文的研究对象为由多个客户端与单个高性能服务器组成的实时系统,客户端与服务器之间通过高速网络连接。客户端向服务器随机发出实时请求,这些请求被服务器视为待处理的实时任务。每个任务的基本属性已知,且到达时间随机。实时任务按照时间关键程度不同分为关键任务和普通任务,其中关键任务必须在时限前完成,而普通任务的运行违限可被放弃。这类实时系统的调度问题可以归结为: 如何调度每个新到达的实时任务到服务器上运行,使得该任务的实时性和安全性约束得到满足。

1.2 实时调度模型

实时调度模型包括 3 个部分: 实时任务模型,系统安全模型和容错任务模型。

1.2.1 实时任务模型

假设客户端发出的多个实时请求可视为一组非周期的独立实时任务集 $\Gamma = \{\tau_1, \tau_2, \cdots\}$,对于实时任务 $\tau_i \in \Gamma$ 用多元组来表示($C(\tau_i)$,$D(\tau_i)$,$R(\tau_i)$,$L(\tau_i)$,$S(\tau_i)$,$B(\tau_i)$) ,其中 $C(\tau_i)$ 表示任务 τ_i 的最大运行时间,$D(\tau_i)$ 表示任务 τ_i 的时限(deadline) ,$R(\tau_i)$ 表示任务 τ_i 的响应时间,即从该任务的到达直至运行完成的时间间隔。$L(\tau_i)$ 表示任务 τ_i 的剩余运行时间,$S(\tau_i)$ 表示任务 τ_i 的已启动时间,即从任务到达直至当前时刻的时间间隔,$B(\tau_i)$ 表示任务 τ_i 被高优先级任务抢占的时间。$State(\tau_i)$ 表示任务的当前状态,包括正在运行(running) ,等待运行(waited) ,被阻塞(blocked) ,被丢弃(discarded) ,已完成(finished) 等类型。实时系统的处理机用 P 来表示,而 τ_{running} 表示处理机上正在运行的任务。

关键任务用 τ_{ci} 来表示,而普通任务用 τ_{ni} 来表示。本文如无特别说明,τ_i 可泛指一切实时任务。系统为每个实时任务 τ_i 指定惟一的优先级,记为 $\Phi(\tau_i)$,所有优先级高于任务 τ_i 的任务集合记为 $hp(\tau_j \mid \Phi(\tau_i), \tau_j \in \Gamma\}$,而所有优先级低于任务 τ_i 的任务集合记为 $lp(\tau_i) = \{\tau_j \mid \Phi(\tau_j) < \Phi(\tau_i), \tau_j \in \Gamma\}$ 。所有关键任务的优先级均高于普通任务,即 $\forall \tau_{ci} \in \Gamma, \tau_{nj} \in \Gamma, \Phi(\tau_{ci}) > \Phi(\tau_{nj})$ 。

1.2.2 系统安全模型

系统安全模型分为 3 个部分: 安全服务、系统安全级别和安全策略。

假设实时系统提供了 K 种安全服务,定义为 $SV = \{sv_1, sv_2, \cdots, sv_K\}$ 。对于每种安全服务 sv_k ,系统提供了一个服务实现集合用 $SS_k = \{s_1^k, s_2^k, \cdots\}$ 来表示,其中 $s_t^k \in SS_k$ 表示安全服务 sv_k 的一个服务实现,$QoS(s_t^k)$ 表示该服务实现的安全强度或安全质量,而 $C(s_t^k)$ 表示该服务实现被调用的单位时间开销。为了叙述方面,本中将每个服务实现集合 SS^k 的元素按照安全强度递增排序。

假设实时系统制订了 P 种安全级别,用 $SL = \{sl_1, sl_2, \cdots, sl_P\}$ 来表示,其中 sl_j 的值越大,表示系统安全级别越高。当系统安全级别为 sl_j 时,系统将每种安全服务 sv_k 的服务实现集合 SS^k 限制为一个子集 $SS^k(sl_j)$,进入调度的实时任务只允许从 $SS^k(sl_j)$ 挑选合适的服务实现来调用。

安全策略(security policy) 是针对不同系统安全级别而制定的具体解决方法,它规定了实时任务选用何种安全服务以及该类服务采用何种服务实现。假设当前系统安全级别为 sl_j ,对于实时任务 τ_i ,其安全策略可以表示为 $SP(\tau_i, sl_j) = \{s_{i,j}^1, s_{i,j}^2, \cdots, s_{i,j}^K\}$,其中 $s_{i,j}^k$ 表示 τ_i 从服务实现集合 $SS^k(sl_j)$ 中挑选的一个服务实现。

1.2.3 容错任务模型

为了实现容错功能,实时系统采用主副本备份技术,每个关键任务 τ_{ci} 设立一个备份任务 $\tau_{bi} = Backup(\tau_{ci})$,所有的备份任务属于备份任务集合 Γ_{backup} ,其基本属性与关键任务 τ_{ci} 相同,但优先级低于 τ_{ci} ,即 $\Phi(\tau_{ci}) > \Phi(\tau_{bi})$ 。

2　自适应实时容错调度算法 AFTS

基于上述模型,本文提出一个满足动态安全需求的自适应实时容错调度算法 AFTS(Adaptive Fault-Tolerant Scheduling Algorithmn) ,本节首先介绍算法的相关概念,然后给出详细的算法设计。

2.1　相关概念

为了更好地描述算法内容,这里先给出相关定义与定理。

定义 1　对于实时任务 τ_i,当系统安全级别为 sl_j 时,采用安全策略 $SP(\tau_i, sl_j) = \{s_{i,j}^1, s_{i,j}^2, \cdots, s_{i,j}^K\}$,该安全策略产生的安全开销 SO_i 和所达到的安全强度(或安全质量) $QoS_{i,j}$ 可分别通过公式(1) 和(2) 来计算。

$$SO_i = \sum_{k=1}^{K} (L^k(\tau_i) \times C(s_{i,j}^k)) \tag{1}$$

$$QoS_{i,j} = \sum_{k=1}^{K} (\omega_k \times QoS(s_{i,j}^k)) \tag{2}$$

式中 $L^k(\tau_i)$ 表示任务 τ_i 调用安全服务 sv^k 的次数或者处理数据的大小,$C(s_{i,j}^k)$ 和 $QoS(s_{i,j}^k)$ 分别表示服务实现 $s_{i,j}^k$ 的单位计算时间开销和安全强度。ω_k 表示安全服务 sv^k 在系统整体安全性中所占的权重,满足 $\sum_{k=1}^{K} \omega_k = 1$。当 ω_k 的值越大,表示安全服务 sv^k 对系统整体安全性的影响越大。

性质 1　对于实时任务 τ_i,当系统安全级别为 sl_j 时,该任务的安全性需求得到满足,当且仅当该任务采用的安全策略满足公式(3) 。

$$\forall k \in [1, K], s_{i,j}^k \in SS^k(sl_j) \tag{3}$$

定义 2　调度队列(scheduled queue) 上所有的实时任务组成了一个任务集合,称为可调度任务集,记为 $\Gamma_{\text{scheduled}}$。

定义 3　对于实时任务 $\tau_i \in \Gamma \cup \Gamma_{\text{backup}}$,该任务加入到任务集 $\Gamma_{\text{scheduled}}$ 中,其最大响应时间 $R(\tau_i)$ 可通过公式(5) 来计算。

$$R(\tau_i) = S(\tau_i) + L(\tau_i) + B(\tau_i) \tag{4}$$

$$L(\tau_i) = C(\tau_i) + SO_i - C_{\text{used}}(\tau_i) \tag{5}$$

$$B(\tau_i) = \sum_{\tau_j \in hp(\tau_i, \Gamma_{\text{backup}})} L(\tau_i) \tag{6}$$

$$S(\tau_i) = t_{\text{now}} - s_i \tag{7}$$

式中 t_{now} 表示当前的系统时刻,s_i 表示任务 τ_i 的到达时刻,$C_{\text{used}}(\tau_i)$ 表示任务 τ_i 的已运行时间。

性质 2　对于实时任务 $\tau_i \in \Gamma$,该任务加入到任务集 $\Gamma_{\text{scheduled}}$ 中,如果满足以下公式(8) 和公式(9) ,则称该任务在任务集 $\Gamma_{\text{scheduled}}$ 中可被调度。

$$\forall \tau_j \in lp(\tau_i, \Gamma_{\text{scheduled}}) \cup \{\tau_i\}, R(\tau_j) \leq D(\tau_j) \tag{8}$$

$$\forall \tau_{bj} \in lp(\tau_i, \Gamma_{\text{scheduled}}), R(\tau_{bj}) \leq L_{\min}(\tau_{cj}) = D(\tau_{cj}) - C(\tau_{cj}) \tag{9}$$

性质 3　对于备份任务 $\tau_{bi} \in \Gamma_{\text{backup}}$,该任务加入到任务集 $\Gamma_{\text{scheduled}}$ 中,如果满足以下公式(10) 和公式(11) ,则称该任务在任务集 $\Gamma_{\text{scheduled}}$ 中可被调度。

$$\forall \tau_j \in lp(\tau_{bi}), R(\tau_j) \leq D(\tau_j) \tag{10}$$

$$\forall \tau_{bj} \in lp(\tau_{bi}) \cup \{\tau_{bi}\}, R(\tau_{bi} \leq L_{\min}(\tau_{cj}) = D(\tau_{cj}) - C(\tau_{cj}) \tag{11}$$

定义 4　对于实时任务 $\tau_i \in \Gamma \cup \Gamma_{\text{backup}}$,该任务加入到任务集 $\Gamma_{\text{scheduled}}$ 中,如果该任务导致部分普通任务无法被调度,则称这些普通任务为受压制任务,其任务集合记为 $\Gamma_{\text{un-schedulable}}(\tau_i)$。

2.2　算法设计

AFTS 算法分为 3 个部分: SAC(Scheduling Admission Check) 算法, SPA(Security Policy Assign) 算法和 FTPS(Fault - Tolerant Preemptive Schedule) 算法,下面分别进行介绍。

2.2.1　SAC 算法

SAC 算法的核心思想是从到达队列(Arrived Queue) 中循环取出新到达的实时任务,并按照任务类型做出相应处理: 对于关键任务和备份任务,首先判断其是否能够满足可调度性,如果能够满足调度条件,则将其放入调度队列(Scheduled Queue) 。如果该任务不可调度,则以牺牲受压制的普通任务为代价来保证其可调度运行。对于普通任务,只需判断其是否满足可调度条件,将满足条件的任务放入调度队列,丢弃不可调度的任务。算法的基本步骤如下所示。

Algorithm 1. SAC(Scheduling Admission Check)

for each task τ submitted to the arrived queue do

　update scheduling information in $\Gamma_{\text{scheduled}}$;

　set $SL(\tau_i) = sl_j$;

　set $SP(\tau_i, s_{i,j}^k) = \{s_{i,j}^1, s_{i,j}^2, \cdots, s_{i,j}^K\}$, $s_{i,j}^k$ is the first element in $SS^k(sl_j)$;

　switch case $\tau_i \in \Gamma_{\text{critical}} \cup \Gamma_{\text{backup}}$

　　if τ_i is schedulable in $\Gamma_{\text{scheduled}}$ then

　　set $State(\tau_i) = WaitedForPolicyAdjust$;

else if $\Gamma_{\text{un-schedulable}}(\tau_i) \neq \emptyset$

reject every $\tau_i \in \Gamma_{\text{un-schedulable}}(\tau_i)$ to the discarded queue and set $State(\tau_i) = WaitedForPolicyAdjust$;

else

report error of scheduling critical or backup task;

break;

end if

insert τ_i to $\Gamma_{\text{scheduled}}$ by its priority order;

end case

case $\tau_i \in \Gamma_{\text{normadl}}$

if τ_i is schedulable in $\Gamma_{\text{scheduled}}$ then

set $State(\tau_i) = WaitedForPolicyAdjust$;

insert τ_i to $\Gamma_{\text{scheduled}}$ by its priority order;

else

reject τ to the discarded queue;

end if

end case

end switch

end for.

2.2.2 SPA 算法

SPA 算法的核心思想是为调度队列中的所有新加入的任务设置最优的安全策略,安全策略的选择必须满足系统安全级别的限制。算法的基本步骤如下所示。

Algorithm 1. SPA(Security Policy Adjust)

foreach task τ_i with $State(\tau_i) = WaitedForPolicyAdjust$ do

update scheduling information in $\Gamma_{\text{scheduled}}$;

set $sl_j = SL(\tau_i)$;

do

save $SP_{\text{temp}} = SP(\tau_i, sl_i)$;

set $SP(\tau_i, s_{i,j}) = \{s_{i,j}^1, s_{i,j}^2, \cdots, s_{i,j}^K\}$, $s_{i,j}^k$ is set to next element in $SS^k(sl_j)$;

while (τ_i is schedulable in $\Gamma_{\text{scheduled}}$)

set $SP(\tau_i, sl_j) = SP_{\text{temp}}$;

set $State(\tau_i) = ready$;

end for.

2.2.3 FTPS 算法

FTPS 算法分为 2 个并行部分:①按照优先级抢占策略来选择任务到处理机中运行;②根据任务运行的结果反馈来决定是否启动或终止备份任务。算法的基本步骤参见文献[7,8]。

3 仿真实验

本文主要通过 1 组仿真实验来评估 AFTS 算法的性能。为了更好地比较算法的各项性能指标,这里选取文献[6]提出的 SAEDF 算法和文献[6]提出的 TPSS 算法作为本次仿真实验的参照算法。

3.1 性能评价指标

仿真实验采用 3 个性能评价指标:任务的可调度率 GR(Guarantee Ratio),关键任务的运行成功率 SRR(Success Running Ratio),以及系统整体安全性能 GSP(Global Security Performance),其定义如公式(12)~(14)所示。

$$GR = \frac{N(\text{scheduled tasks})}{N(\text{total tasks})}. \tag{12}$$

$$SRR = \frac{N(\text{scheduled and finished tasks})}{N(\text{total tasks})} \tag{13}$$

$$GSP(sl_j) = \frac{\sum_{\tau_i \in \Gamma_{\text{scheduled}}} (\Phi\tau_i) \times OoS(\tau_i, sl_j))}{\sum_{\tau_i \in \Gamma_{\text{scheduled}}} \Phi(\tau_i)} \tag{14}$$

式中 GR 表示任务调度成功率,GR 的值越大,调度策略就越有效。SRR 表示关键任务的运行成功率,它也是系统运行可靠性指标,SRR 的值越大,关键任务成功率越高,系统可靠性越高。$GSP(sl_j)$ 表示系统在安全级别为 sl_j 时的总体安全性能,其中 $\tau_i \in \Gamma_{scheduled}$ 表示所有被成功调度到调度队列的实时任务。$GSP(sl_j)$ 的值越大,系统所实现的总体安全性越高。

3.2 实验环境及参数

仿真实验的全部代码采用 C++语言编写,实验环境为 1 台配置为 P4 2.0GHz CPU,4G RAM,Windows XP SP3 的计算机。参与实验的 1 组共 200 个实时任务通过随机方式产生,其中包括关键任务、备份任务和普通任务,其个数比例依据关键任务占比率(Critical Tasks Ratio,即关键任务占全部任务的比例)来调节。系统提供的安全服务的设置参照了文献[6],具体考察 3 种安全服务类型:加解密服务,完整性服务和认证服务,各种服务的安全开销和单位计算时间开销也来自文献[6]。实验将系统安全级别分为 8 个级别,对于每个级别,我们分别定义了对应的服务实现集合。

3.3 实验结果分析

3.3.1　系统安全级别对任务可调度性的影响

这个实验主要考察系统安全级别的动态调整对任务可调度性的影响,实验中关键任务故障率设置为 0,关键任务占比率设置为 20% 。

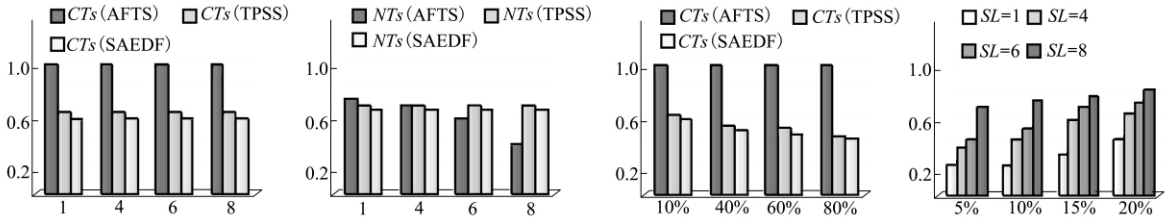
实验结果如图 1a) 和图 1b) 所示,其中横轴表示系统安全级别,纵轴表示任务的可调度率 *GR*、*CTs* 表示关键任务或备份任务,而 *NTs* 表示普通任务。当系统安全级别从 1 提升至 8 时,TPSS 算法和 SAEDF 算法的 2 种任务的可调度率可调度率无明显变化,这是因为这 2 种算法在设计时没有考虑系统安全级别,因而系统安全级别的改变,对这两种算法没有影响。AFTS 算法的关键任务可调度率始终保持在 100% ,而普通任务调度率却随着安全级别的提升而迅速下降,这是因为 AFTS 将关键任务和备份任务的可调度性置于首要地位,当系统安全级别上升时,AFTS 算法将通过牺牲普通任务的可调度率为代价来保证关键任务和备份任务的可调度性。综上所述,AFTS 算法在对系统安全级别的适应性上优于其它算法。

3.3.2　关键任务故障率对关键任务运行成功率的影响

这个实验主要考察关键任务故障率对关键任务运行成功率的影响,实验中系统安全级别设置为 1,关键任务占比率设置为 20% ,实验结果如图 1c) 所示。实验中的关键任务故障通过程序预先设置的方式来模拟,由于 TPSS 和 SAEDF 2 种算法不具有容错功能,我们暂且将任务集中的备份任务视为普通任务来处理。图 1c) 中横轴表示任务发生的故障率,纵轴表示任务的运行成功率 *SRR*,*CTs* 表示关键任务或备份任务,而 *NTs* 表示普通任务。当关键任务故障率从 10% 升至 80% 时,TPSS 算法和 SAEDF 算法的 *CTs* 降低明显,这是因为 2 种算法没有考虑容错功能,发生故障的关键任务得不到有效处理。而 AFTS 的 *CTs* 始终维持在 100% ,这是归功于备份任务对故障的恢复作用。综上所述,AFTS 算法在容错能力上优于其它算法。

3.3.3　系统的整体安全性能评估

这个实验主要考察系统在不同安全级别下的整体安全性能,实验结果如图 1d) 所示。图中横轴表示关键任务占比率,纵轴表示系统的整体安全性能 *GSP*,*SL* 表示系统安全级别。当关键任务占比率固定时,提升系统的安全级别会使得 *GSP* 提升明显,这是因为高系统安全级别下,对任务的安全需求增加,迫使任务采用更高安全强度的服务实现,从而增加了 *GSP*。当系统安全级别固定时,提升关键任务占比率会使得 *GSP* 增加显著,这是因为相同的安全级别下,具有高优先级的关键任务数量越多,*GSP* 也会增加越多。综上所述,实验证明了适当增加系统安全级别和关键任务占比率能够显著提升系统的整体安全性能。



a)　*SL* 对 *CTs-GR* 的影响　　b)　*SL* 对 *NTs-GR* 的影响　　c)　*FR* 和 *SL* 对 *GR* 的影响　　d)　*SL* 和 *CR* 对 *GSP* 的影响

图 1　仿真实验结果

4　结　论

本文针对现有算法不足,提出了一种自适应实时容错调度算法(AFTS) ,该算法支持优先级抢占式调度策略,以牺牲普通任务的运行为代价来保证关键任务的可调度性,并采用主副本备份技术实现了关键任务的容错功能。当系统安全级别动态调整时,该算法能够为实时任务选择满足当前安全需求的最优安全策略。

下一步工作包括: (1) 将算法推广到分布式实时领域,支持多处理机调度; (2) 构建更为精确的安全开销模型。

参考文献:

[1] 熊光泽，常政威，桑 楠. 可信计算发展综述. 计算机应用，2009，29(4)：915-919
Xiong Guangze，Chang Zhengwei，Sang Nan. Survey on Dependable Computing. Journal of Computer Applications，2009，29(4)：915-919 (in Chinese)

[2] Liu C L，Layland J W. Scheduling Algorithms for Multiprogramming in a Hard Real-Time Environment. Journal of the ACM，1973，20(1)：46-61

[3] Kalogeraki V，Melliar-Smith P M，Moser L E. Dynamic Scheduling for Soft Real-Time Distributed Object Systems. The 3rd IEEE International Symposium on Object-Oriented Real-Time Distributed Computing，2000，114-121

[4] Luis Almeida. A Dynamic Scheduling Approach to Designing Flexible Safety-Critical Systems. Journal of the ACM，2003，22(4)：46-61

[5] 朱晓敏，陆佩忠. 异构集群系统中安全关键实时应用调度研究. 计算机学报，2010，33(12)：2364-2377
Zhu Xiaomin，Lu Peizhong. Scheduling for Security-Critical Real-Time Applications on Heterogenous Cluster. Chineses Journal of Computers，2010，33(12)：2364-2377 (in Chinese)

[6] Xie Tao，Qin Xiao. Scheduling Security-Critical Real-Time Applications on Clusters. IEEE Trans on Computers，2006，55(7)：864-879

[7] 夏 平，周兴社，骆万文等. 面向分布式实时系统的新型可信任务调度算法. 西北工业大学学报，2011，29(2)：155-159
Xia Ping，Zhou Xingshe，Luo Wanwen，et al. A New and Better Dependable Task Scheduling Alogrithm for Distributed Real-Time System. Journal of Northwestern Polytechnical University，2011，29(2)：155-159 (in Chinese)

[8] Xia Ping，Zhou Xingshe. Security-Driven Fault Tolerant Scheduling Algorithm for High Dependable Distributed Real-Time System. Proceedings of the 2011 Fourth International Symposium on PAAP，2011：29-33

A Better Adaptive Fault-Tolerant Scheduling(AFTS) Algorithm for Real-Time Tasks with Dynamic Security Requirements

Xia Ping，Zhou Xingshe，Xie Bichang

(Department of Computer Science and Engineering，Northwestern Polytechnical University，Xi′an 710072，China)

Abstract: Current fault-tolerant scheduling algorithms have considered either the dynamic security requirement or the schedulability one of real-time task，but，to our best knowledge，not both together. Sections 1 and 2 of the full paper explain our AFTS algorithm，which we believe is better than existing ones and whose core consists of: "Firstly，it builds a real-time scheduling model，which is adaptive to dynamic security requirement，and the model describes in detail the scheduling process，including real-time tasks，system security services，and task fault-tolerantance. Based on the model，it proposes a new AFTS algorithm which supports priority preemptive scheduling policy，and guarantees the schedulability of critical tasks at the cost of deferring normal tasks and adopts primary/backup copy technique to ensure fault-tolerance of critical tasks. The algorithm selects the best-fit security policy for schedulable task under updated system security level.". Simulation results，presented in Fig. 1，and their analysis show preliminarily that the improvements in the adaptability to dynamic security level，the schedulability，and the fault-tolerance of critical task can indeed be achieved by using our AFTS algorithm.

Key words: algorithms，computer simulation，design，efficiency，mathematical models，real time systems，scheduling，security of data；adaptive fault-tolerant real-time scheduling algorithm，dynamic security requirement