基于对象中间件的自适应容错技术*)

Adaptive Fault Tolerance Technology Based on Object Middleware

李琪林　陈　宇　周明天

(电子科技大学计算机科学与工程学院　成都610054)

Abstract Distributed object-oriented systems are widely applied to mission-critical systems, such as real-time systems, online paying systems and stock exchange systems. Presently, main problem of business distributed object-oriented systems is how to integrate fault-tolerance service with system and still provide correct service for user at the event of system failure. In addition, distributed systems must always operate in highly dynamic environments, so fault tolerance mechanism should provide more intelligence to adapt itself to in response to the changes in system resource, application demands and user requirements. To solve these problems, adaptive fault-tolerance technology based on object middleware isadvanced. This paper analyzes main features of adaptive fault tolerance based on object middleware and its problems, evaluates representative systems, and introduces our present work. Finally we give future research direction.

Keywords Distributed computing environment, Adaptive fault-tolerance, Object middleware, CORBA

一、引言

近年来,计算机系统逐渐渗透到人类生活中的各个方面,对人类生活产生了不可估量的影响。目前,面向对象的分布式系统已经被广泛地应用于关键业系统中,如空中交通管制系统、在线支付系统和核电站控制系统等。显然,提供高可靠的服务是用户对系统最基本的要求,否则将造成重大财产,甚至人员损失。在过去的几十年间,针对分布式系统生命周期的不同阶段,研究人员提出了大量提高系统可靠性的方法。在分布式系统运行过程中,容错技术是保证分布式系统运行可靠性的重要手段。随着硬件可靠性的提高,软件设计错误成为系统的主要错误源,因此软件容错成为决定系统可靠性的极其重要的因素[1]。目前,主要的软件容错模型包括版本复制、RB模型、NVP模型和NSCP模型[2]。然而,由于软件容错实现的复杂性,分布式系统,特别是基于商用平台的分布式应用,出于对成本、可靠性和用户需求的综合考虑,大都仅采用检测点与回滚这一简单交易模式。而在部分专用系统中,设计人员采用版本复制实现容错。

随着分布对象计算技术在关键业务领域的广泛应用,引入更加复杂的容错模式以提高分布式系统的容错性能是大势所趋。如何将容错技术有机地结合在分布式系统中,是实现高可靠的分布式容错系统面临的重要问题[3]。

如前所述,商用分布式系统软件平台仅提供十分有限的可靠性保障机制,需要用户自行构建所需的软件容错结构。由于容错结构自身的高度复杂性,加之用户对系统认识的限制,最终实现的应用系统往往庞大、复杂,反而增加了出错的可能,甚至影响软件平台的正常工作。因此,商用分布式系统软件平台应该具有根据应用的可靠性需求、资源和运行环境的状况,为应用提供构建、管理容错结构的全面服务,实现对于应用的透明容错,因而将用户从实现软件容错的繁琐工作中解放出来,使之能够将主要精力集中于实现应用软件的功能。显然,这需要对软件平台进行改进,增加了平台实现的复杂性,并使之出现从通用平台向专用平台转变的趋势。近年来兴起的对象中间件技术为这一问题提供了新的解决方案。利用对象中间件实现容错一方面避免了修改底层软件平台,同时具有良好的可移植性和可扩展性。

将容错技术与分布式系统有机地结合面临的另一个主要问题是如何选择合适的容错策略。传统的分布式应用将功能与容错作为一个整体实现。因而,采取何种容错策略,在系统设计阶段就已经确定,并在该应用的整个生命周期中保持不变。然而,由于应用存在于高度动态的分布式运行环境中,运行环境、系统资源,甚至应用的容错要求都随着时间的推移而变化,因此仅依靠单一固定的容错策略显然无法适应。这就要求系统平台能够智能地根据外部运行环境和应用需求的变化,选择合适的容错策略,以满足应用可靠性的要求,同时提高系统资源利用率。

基于对象中间件的自适应容错将几种适用于不同运行环境和系统负载情况的容错策略结合起来,根据应用本身的属性和资源状况,动态地为任务选择当前状态下最佳的容错策略,在保证系统可靠性要求的前提下,灵活地配置系统资源,提高资源的利用率和系统的吞吐量。

本文讨论基于对象中间件的自适应容错的基本特征和需要解决的主要问题;介绍典型的基于对象中间件的自适应容错的分布式系统,其中也包括我们目前的工作;指出该领域未来的研究方向;最后,给出了结论。

二、基于对象中间件的自适应容错

容错是指如何保证在出现错误时系统仍能提供正确服务。在硬件可靠性飞速提高的前提下,软件容错成为决定系统可靠性的重要因素。软件容错的基础是冗余,它可以分为软件冗余和时间冗余,即为同一个应用提供多个运行实例,或在时限到来前满足应用的多次运行。软件容错面临的两个主要问

*)本文获四川省重点科技计划项目基金资助。李琪林　博士研究生,主要研究方向:分布式系统,对象中间件技术。陈　宇　博士研究生,主要研究方向:实时操作系统,软件容错技术。周明天　教授,博士生导师,主要研究方向:网络,对象中间件技术,应用服务器。

题是:1)如何在保证系统可靠性的前提下提高系统资源的利用率;2)容错结构如何适应不断变化的应用需求和运行环境。显然,过高的冗余度对资源是一种浪费,而应用软件仅采用一种容错策略也是不合理的。自适应容错正是为解决这一问题而提出的。自适应容错的中心思想是:根据系统的空闲资源现状和应用的容错要求,动态地为应用选择容错策略,构建容错结构,在保证应用软件可靠性的基础上,尽可能提高系统资源利用率。

基于中间件的自适应容错,利用系统平台提供的通用服务,根据应用程序提出的容错要求,通过改进对象中间件,使之在提供标准的服务和协议的同时,实现冗余应用对象的管理和可靠的通信,为应用构建和管理容错结构,满足应用的容错要求。这样,既保证了系统平台的通用性,又降低了应用程序的实现复杂度。因此,在保证系统可靠性的基础上提高系统资源利用率具有重要的理论价值和实际意义。

2.1 基于对象中间件的自适应容错的特点

我们认为,基于对象中间件的自适应容错技术应该具有以下几个主要特征:

高效 自适应容错机制应该能够提高系统的可靠性,同时其运行开销必须保持在可以接受的范围内;

高的资源利用率 自适应容错机制应该通过提高资源的利用率降低系统对冗余资源的需求,并能够保证在部分资源暂时无法使用的情况下,满足关键任务的可靠性要求;

通用性 自适应容错机制必须能够应付不同类型的系统错误,如外部设备错误、处理器错误、系统软件和应用软件错误,同时也应该能够处理暂时错误和永久错误;

可配置 自适应容错机制必须能够接受通过参数形式传递的应用程序对于容错的特殊要求,并基于这些参数进行容错策略的选择;

多样性 自适应容错机制应该包含多种不同的容错策略,以响应运行环境的变化;

界面友好 自适应容错机制作为分布式系统的一个模块,应该具有简单、清晰的编程接口,以减轻应用程序设计者的负担;

验证简单 将过去对整个系统进行验证以发现问题,保证系统的容错功能,改变为较小的自适应容错系统以保证容错功能的正确实现。

2.2 关键技术

自适应容错的主要功能是为应用提供动态的容错管理支持。当系统需要创建一个具有容错需求的应用实例时,自适应容错首先获取该实例的容错决策相关信息,并通过中间件支持获得当前系统资源的可用信息,基于以上信息,利用自适应算法为该实例选择合适的容错策略。在容错策略确定后,自适应容错为应用实例构建并管理容错结构。为了实现以上功能,自适应容错必须解决如下关键技术。

容错决策 容错决策为应用实例确定合理的容错策略。不同的容错策略在资源需求、响应时间和复杂度方面有很大区别,因而适用于不同的应用场合。例如,NVP 和 DRB 等采用冗余实例的并行运行,具有确定的响应时间,容错算法简单,但系统资源需求量大。RB 采用冗余实例的串行运行,响应时间则因为冗余实例是否出错而有很大波动,容错算法复杂,但资源需求较小。同时,容错决策必须确定应用实例的冗余级别,对于 NVP 而言,$n \geq 2f+1$,对于 RB 而言,$n \geq f+1$其中 n 为冗余级别,f 为应用实例可以容忍的出错冗余实例数。容错决策根据应用的可靠性需求确定其容错策略和冗余级别。容错决策是自适应容错的核心,是自适应容错是否容错的决定因素。容错决策需要对象中间件的资源管理功能的支持。

冗余组的结构管理 当应用实例的容错策略和冗余级别确定后,自适应容错构建相应的冗余组,创建规定数量的组成员,并使之协调工作。冗余组对外表现为一个独立的进程,具有统一的接口,向外提供具有确定质量的服务,而其内部结构是透明的。在冗余组的运行过程中,结构管理根据组成员的加入或退出更新组结构,保证冗余组继续正常工作。冗余组结构的管理涉及网络的监控和重配,目前有三类基本的方法:集中式、分散式和混合式[4]。集中式包括监控节点方案[5]和其它的方案[6];分散式包括 PRHB(Periodic Reception History Broadcast)方案[7,8]和 TTP(Time-Triggered Protocol)方案[9];混合式包括 SNS(Supervisor-based Network Surveillance)方案[10]及其它的一些解决方案。其中 SNS 方案在具有不同拓扑结构的点对点网络中尤其有效。冗余组结构的管理可以通过对象中间件提供的服务加以实现。

可靠的组通信 组通信是冗余组正常运转的重要保证,而建立组通信面临的最主要的问题是应对可能出现的通信错误[11]。在点到点的分布式运行环境中,组通信是一组点到点通信的集合,而在一个具有广播能力的分布式运行环境中,可以通过将组 ID 加入消息头中,通过一次广播实现组通信。可靠的组通信必须达到:在确定的时限内,所有的接收者均正确地获得消息,或放弃该次广播[12]。可靠的组通信依赖于对象中间件提供的通信服务。

组内一致 冗余组中所有成员必须按照同一顺序接收和处理外来信息,以保证其运行结果的一致性。然而,由于所处物理位置和其他条件限制,冗余组成员收到外部输入的时间顺序可能不同。因此,自适应容错必须对外部输入进行排序,并控制各成员按照同样的顺序进行处理。此外,同一外部输入到达各成员的时间延迟必须在规定范围内,以保证用户对应用实例响应时间的要求[13~17]。

运行环境监测 监测运行环境的动态变化是自适应容错实现各项管理功能的基础。自适应容错只有在正确地了解当前系统运行环境的前提下,才能正确地进行容错决策、容错结构管理和可靠的组通信[4,18~20]。

实际上,上述各项关键技术在分布式系统中已分别得到较为深入的研究,并且均存在多种不同实现方法。因此,自适应容错的实质在于选择合适的实现机制,将各种关键技术有机地结合起来,形成一个整体,向用户提供透明的容错支持,在保证应用可靠性的基础上,智能地管理系统资源,提高了系统资源利用率,降低了系统成本。在第三节中,我们分析几种典型对象中间件自适应容错的系统结构,及其采用的相关技术。

三、典型分布式自适应容错系统的概述和评估

二十世纪九十年代的前期,有关自适应容错的研究主要围绕其基本概念和实现机制进行,提出了自适应容错的概念框架[21~23]。Kim 和 Lawrence 对自适应容错在复杂的实时分布式应用方面进行了分析,并提出了 action-level 容错[24],Bondavalli 等在分布式实时操作系统 Spring 上提出了自适应容错模型 FERT[25]。目前,国内外一些研究机构陆续建立了一些具有自适应容错功能的实验系统,正在逐步走向实用。以下,我们将简要分析典型的对象中间件自适应容错系统 Pro-

teus[26]、Chameleon[27]和 Tong-AFT[28]的系统结构和关键技术。

3.1 AquA/Proteus

AQuA 是一个操作系统体系结构，为分布式容错系统提供灵活的基础设施。AQuA 在 CORBA 的支持下，提供动态的对象拷贝，满足用户的可靠性要求。Proteus[26]是 AQuA 系统的核心部分，它通过自适应重配提供动态容错。应用服务质量对象(Quality Object，简称 QuO)，Proteus 允许用户指定他们要求的可靠级别。QuO 运行体负责与 AQuA 交互，检测和处理错误。

3.1.1 *系统结构* Proteus 分为三部份：可靠性管理器(Dependability Manager，简称 DM)、对象工厂(Object Factory，简称 OF)和网关(Gateway)。通过 QuO，运行体与 AQuA 连接。DM 又包括决策支持顾问(Advisor)和协议协调器(Protocol Coordinator，简称 PC)。决策支持顾问根据系统信息和应用要求，进行系统决策并确定错误处理的策略。在 PC 的支持下，Advisor 的请求被传送到 Proteus 的其它部分。PC 使用 Ensemble 组通信系统保证组间可靠的消息传递。

对象工厂在每一本地节点实例化，它负责监控、创建和杀死本地进程。Advisor 发起对拷贝进程的请求，对象工厂提供进程的实例化和注册。对象工厂维护所有本地进程的信息和组结构。网关部分提供关键的组管理基础设施。

3.1.2 *可靠的组通信和组内一致机制* Gateway 实现了四类组通信服务：管理拷贝、连接、点对点和广播。Gateway 通过提供选举机制保证组间的一致性并通过主组将结果送往所有的节点。每一个组都指定了一个主进程发送消息，提供可靠的多播通信，在 Ensemble 的支持下，保证操作的原子性。进程间的同步通过主进程和从进程间的消息发送和确认维护。

由于既可能主进程也可能从进程出错，因此一旦主进程检测到从进程无法在超时到来之前送回确认信息，或主进程发现某进程状态出错，则它认为检测到从进程出错。若主进程失败，则发现主进程失败的从进程成为新的主进程，同时修改系统信息。

Proteus 系统提供强大的组通信服务，支持可靠的组通信，保证组内一致性。它将自适应过程可视化，方便用户管理。同时针对不同的可靠性要求，Proteus 系统提供不同级别的可靠性满足不同应用的需求。但是，Proteus 系统没有提供环境监测机制，因此无法动态地响应外部环境的变化，必须通过 QuO 运行实体，由用户手工注册额外的资源到 Advisor。另外，Proteus 未提供实时服务，不能支持关键业务应用。

3.2 Chameleon

Chameleon[27]是一个可重配的软件层，它应用 ARMORs (Adaptive, Reconfigurable, Mobile Objects for Reliability)提供容错和自适应决策。ARMOR 由基本的对象组成，这些对象对应低层的实现机制例如 MessageSend 对象和 MessageDispatch 对象，负责发送消息到 ARMOR 和接收来自 ARMOR 的消息。这些基本对象支持独立于主机的原语，定义 ARMORs 的功能。

3.2.1 *系统结构* Chameleon 系统分为两层。上层是动态的 ARMORs 对象集，可以在联网系统中分发和复制。低层是 Chameleon 的 API，为 ARMORs 对象提供支持及引导来自应用的调用。这两层均由操作系统提供支持。

Chameleon 通过子类支持实时应用。容错管理器(Fault-Tolerance Manager，简称 FTM)子类被实例化，提供实时操作。实时管理器(Real-Time Manager，简称 RTM)提供与 FTM 类似的功能，只是它由不同的基本对象组成。与普通基本对象相比，构成实时操作的基本对象具有确定的延迟。应用这些基本对象，可以建立所有 ARMORs 子类，保证可以预测的执行时间。

3.2.2 *可靠的组通信和组内一致机制* Chameleon 为用户提供语言设计容错执行策略(Fault-Tolerance Executive Strategy，简称 FTES)。这些策略详细描述了组成员的精确数及分布情况。FTM 负责维护这些策略的注册。根据性能和可靠性要求，可以调整这些策略。例如，从性能的角度考虑，组成员产生的第一个结果可以返回或通过选举机制，与输出结果达成一致(从可靠性角度考虑)。

当组成员出错时，可能要求增加新成员和保持组内状态的一致性，这由 FTES 实现。这时可能会出现以下几种情形。代理管理器(Surrogate Manager，简称 SM)负责协调恢复。恢复的技术包括进程迁移、应用重启或增加新的组成员。状态传递是最后的选择，由当前组成员的检测点拷贝实现。

守候进程(Daemon)负责组通信。Daemon 安装在每个节点并至少实现一种网络协议。所有节点内的通信先送往 Daemon，再从 Daemon 发送到目的地。节点内通信由进程间通信设施实现。节点间通信通过 Daemon 从发送节点传递到接收节点。

Fanout ARMOR 提供原子的多播通信和所有消息的订阅设施保证组成员同步。不同的节点有不同的伪随机数生成器和算法，通过协调，单一的随机数被广播到所有的组成员，生成有意义的结果。

3.2.3 *环境监测机制* Chameleon 通过超时和依赖关系实现错误检测。依赖关系指定系统的哪部分负责检测其它部分的错误。依赖是环形的(cyclic)，可以保证发现的错误信息向后传递，通知系统的其它部分。偶发性错误、崩溃和时间错误由每个正常的操作检测。

当在 FTES 中指定错误时，可以保证应用的绝大部分达到预期的行为。特别地，当能从某些组成员获得有意义结果时，可以实现透明性。当组成员返回不同的结果时如结果值出错，Voter ARMOR 可以实现用户指定的选举策略。当至少有一个组成员能提供服务时，崩溃和偶发性错误可以被屏蔽。在检测点 ARMOR(Checkpoint ARMOR)的帮助下，可以为应用屏蔽失败，并在检测点操作完成后，将结果返回给用户。

Chameleon 层次结构清晰，功能强大。利用 ARMORs 实现容错和提供自适应决策。它为用户提供相应的语言设计容错执行策略，并在守候进程的支持下，保证可靠的组通信和组内一致性。另外，Chameleon 提供了环境检测机制，能有效地检测环境的变化，实现自适应容错。但是，Chameleon 强烈地依靠操作系统的支持，因此系统的可移植性差；同时，它必须由应用支持，实现组成员同步，将本应由系统实现的功能转移给应用开发者，这无疑加重了应用开发者的负担。另外，尽管 Chameleon 提供了环境监测，但其很大程度上取决于自适应策略的具体表示，因此环境监测方法显得不够灵活。

3.3 Tong-AFTS

近年来，以 CORBA 为代表的分布对象技术日益成熟，越来越多的分布式系统基于 CORBA 构建。CORBA 是分布对象计算的中间件标准，为分布应用的开发和布署提供标准的服务和协议。基于 CORBA 实现的自适应容错模型可以充分

利用 CORBA 提供标准服务和协议，简化应用对象间的通信；同时，CORBA 先天的平台无关性方便系统的移植；为此在跟踪国外技术发展的基础上，我们提出了基于 CORBA 的分布式系统自适应容错模型及其组成部分的实现策略，并在 Windows NT 和我们自行研制的遵循 CORBA 规范的对象中间件平台 TongBroker 上实现了 Tong-AFTS[28]（TongBroker-based Adaptive Fault-Tolerance System）系统。

3.3.1 *系统结构* Tong-AFTS 系统包括自适应管理器、容错策略管理器和系统监控管理器。它利用底层操作系统和中间件提供的服务，进行资源分配，支持应用间通信，监控环境、用户和应用的变化，根据用户和应用的要求，提供相应的容错机制，实现应用的自适应容错。Tong-AFTS 系统各部分的交互关系如图1所示。

自适应管理器是 Tong-AFTS 系统的核心，它根据环境和用户需求变化，为应用选择最合适的容错结构。自适应管理器做出的自适应决策转发给容错策略管理器，由容错策略管理器为应用分配资源以适应满足应用需求。系统监控管理器维护系统的当前状态信息，监控分布式系统中各节点的状态。

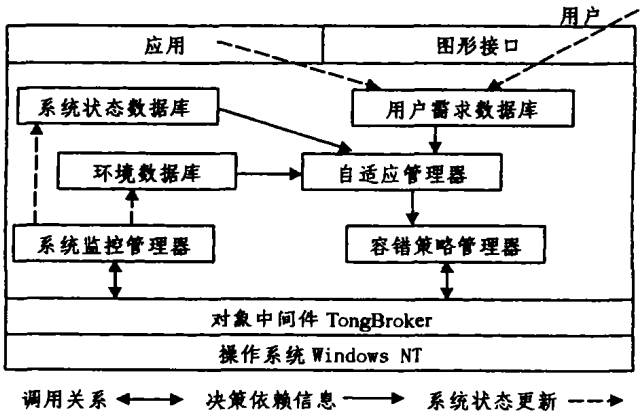

图1 Tong-AFTS 各部分的交互关系

3.3.2 *容错决策* 容错策略管理器根据自适应管理器的自适应决策，提供不同的容错方案，支持广泛的应用。容错策略管理器主要提供用户透明的自适应容错服务。目前，容错策略管理器支持三类容错方案 P-E、RB 和 DRB。

为了支持自适应容错，Tong-AFTS 系统需要维护以下三个数据库：

1）环境数据库：记录分布式环境中远地节点的状态信息。系统监控管理器负责监视环境变化，维护和更新数据库。

2）内部状态数据库：记录本地节点的资源状况。系统监控管理器负责监视环境变化，维护和更新数据库。

3）用户需求数据库：记录用户为应用定义的特殊属性，如应用的重要性。

3.3.3 *运行环境的监测方法* 系统监控管理器维护系统的当前状态信息。它主要负责监控分布式系统中各节点的状态。当远地节点无法正常工作时，系统监控管理器将该信息记录在环境数据库中，供自适应管理器和容错策略管理器使用。当节点从失败中恢复或新的节点加入时，系统监控管理器会更新环境数据库，便于正常节点的快速学习。另外，系统监控管理器从底层的资源管理模块中获取本节点的资源信息，该信息记录在内部状态数据库中，供自适应管理器和容错策略管理器使用。

四、未来的研究方向

基于对象中间件的分布式自适应容错系统的提出，使容错操作对应用透明，减轻了应用程序设计者的工作负担。更重要的是，自适应容错在保证应用可靠性的基础上，智能地管理系统资源，提高了系统资源利用率，降低了系统成本。但是，引入自适应容错实际上是将容错管理的复杂性转嫁到自适应容错中，因此保证自适应容错系统的可靠性是这一技术成功的关键。未来自适应容错系统需要解决的问题包括引入更加智能的容错决策算法；实时地调整应用的容错结构；提供高可靠、低延迟的组通信手段；实时的系统资源监测；设计更加合理的系统结构[12]等。

另外，分布式自适应容错系统应能动态地检测资源的变化，而目前的大多数系统仅能处理资源失败和资源不可用。当系统增加新的资源时，需要用户通知系统。因此我们认为，如何有效地检测环境变化，提供更好的自适应决策，也是分布式自适应容错系统未来的一个重要研究方向。目前，分布式系统在可靠的组通信、同步等方面已相当成熟，为自适应容错的进一步发展提供了技术上的保证。

结论 基于中间件的自适应容错，利用系统平台提供的通用服务，根据应用程序提出的容错要求，通过改进对象中间件，使之在提供标准的服务和协议的同时，实现冗余应用对象的管理和可靠的通信，为应用构建和管理容错结构，满足应用的容错要求。

本文回顾了基于中间件的自适应容错的发展历史，总结了它的技术特点，分析了它的关键技术，并且简要描述了一些典型的实验系统及其关键技术的实现方法。我们认为，随着系统结构和关键技术的逐渐发展成熟，基于对象中间件的自适应容错作为提高分布式系统可靠性的十分重要的手段，必将在高可靠的分布式系统中发挥更大的作用。

参考文献

1 Kim K H, Lawrence T. Adaptive Fault-Tolerance in Complex Real-Time Distributed Application. Computer Communication, 1992, 15(4): 243～251

2 Cristian H. Understanding fault-tolerant distributed systems. Communications of the ACM, 1991, 34(2): 56～78

3 Lyu M R. Software Fault Tolerance, Wiley, 1995

4 Shokri E, et al. Architecture of ROAFTS/Solaris: A Solaris-Based Middleware for Real-Time Object-oriented Adaptive Fault Tolerance Support. In: Proc. COMPSAC'98 (22 nd IEEE CS Software & Applications Conf.) Vienna, Austria, Aug. 1998. 90～98

5 Hetch M, et al. A Distributed Fault Tolerant Architecture for Nuclear Reactor and Other Critical Process Control of Applications. In: Proc. IEEE CS 21st Int'Symp. On Fault-Taulerance Computing, Montreal, June1991. 462～469

6 Garcia-Molina H. Elections in a distributed computing system. IEEE Trans. On Computers, Jan. 1982. 48～59

7 Kopetz H, Grunsteidl G, Reisinger J. Fault-Tolerant Membership Service in a Synchronous Distributed Real-time System. In: Proc. IFIP WG 10.4 Int'l Working Conf. on Dependable Computing for Critical Applications, Santa Barbara, Aug. 1989. 167～174

8 Kim K H, Shokri E H. Minimal-Delay Decentralized Maintenance of Processor-Group Membership in TDMA-Bus LAN Systems. In: Proc. IEEE CS 13th Int'l Conf. on Distributed Computing Sys-

tems, Pittsburg, May 1993. 410～419
9 Kopetz H, Grunsteidle G. TTP-A: Time-Triggered Protocol for Fault Tolerant Real-Time Systems. In: Proc. IEEE CS FTCS-23, Toulose, France, June 1993. 524～533
10 Kim K, Subbaraman C. A Supervisor-Based Semi-Centralized Network Survillance Scheme and the Fault Detection Latency Bound. In: Proc. 16th Symp. on Reliable Dist. Systems, Oct. 1997. 146～155
11 Kim K. Group Communication in Real-Time Computing Systems: Issues and Directions. In: Proc. FTDCS'99 (7th IEEE Workshop on Future Trends of Distributed Computing Systems), Cape Town, South Africa, Dec. 1999. 252～258
12 Kim K. Issues Insufficiently Resolved in Century 20 in the Fault-Tolerance Distributed Computing Field. In: Proc. SRD2000(IEEE CS 19th Symp. On Reliable Distributed Systems), Nuremberg, Germany, Oct. 2000. 106～115
13 Barcellos Marinho P, Ezhilchelvan Paul D. A End-to-End Reliable Multicast Protocol Using Polling for Scalability: [Tech. Rept. 609]. Dept. of Computing Science, Univ. of Newcastle upon Tyne, 1997
14 Birman K , Schiper A, Stephenson P. Lightweight Causal and Atomic Group Multicast. ACM Trans. On Computer Systems, 1991, 9(3): 272～314
15 Chang J M, Maxemchunk N. Reliable Broadcast Protocols. ACM Trans. On Computer Systems, 1984, 2(3): 251～273
16 Christian F, Aghili H , Strong R, Dolev D. Atomic Broadcast: From simple message diffusion to Byzantine Agreement. In: Proc. FTCS 15, Ann Arbor, Michigan, 1985. 200～206
17 Cheriton D R, Skeen D. Understanding the Limitations of Causally and Totally Ordered Communication. In: 14th ACM Symposium on Operating Systems Principles, pp. 44～57
18 Shokri E, et al. An Approach for Adaptive Fault Tolerance in Object-Oriented Open Distributed Systems. International Journal of Software Engineering and Knowledge Engineering, 1998, 8(3)
19 Hurley P , Dussault J L. The Development and Assessment of An Adaptive Fault Resistant System(AFRS). In: Proc. of Second International Command & Control Research & Technology Symposium, Warwickshire, UK, Sept. 1996. 256～267
20 Shokri E, Hecht M. Adaptive Fault-Tolerance for Autonomous Spacecraft. NASA SBIR Phase 1, Final Report, SoHaR Inc., Beverly Hills, CA, June 1996
21 Gupta, et al. Adaptive Fault Resistant System(AFRS)", Rome Laboratory USAF: [Technical Report RL-TR-95-3]
22 Goldberg J. Adaptive Fault Resistant System", SRI International: [Technical Report SRI-CSL-95-02]
23 Bihari B, Schman K. Dynamic Adaptation of Real-Time Software. ACM Transactions on Computer Systems, 1991, 9: 143～174
24 Kim K. Action-level fault tolerance, in Advances in Real-Time Systems. S. H. Sang ed., Prentice Hall, 1994. 415～434
25 Bondavalli A, Stankovic J, Strigini L. Adaptive Fault Tolerance for Real-Time Systems, in Responsive Computer Systems: Steps toward Fault-Tolerant Real-Time System. D. S. Fussell and M. Malek ed., Kluwer, 1995. 187～208
26 Sabnis C, Cukier M, Ren J, et al. Proteus: A Flexible Infrastructure to Implement Adaptive Fault-Tolerance in AquA. In: Proc. of the 7th IFIP IWC in DCCA, 1999. 137～156
27 Kalbarczyk Z, et al. Chameleon: a Software infrastructure for Adaptive Fault Tolerance. IEEE Transactions on Parallel and Distributed Systems, 1999, 10(6): 560～579
28 李琪林，陈宇，周明天. 基于CORBA的分布式系统自适应容错模型的研究. 计算机科学，2002，29(3)：119～121

(上接第120页)

÷2次，它大致为$2^{244}\times 2^{16}=2^{260}$。若计算机每秒能做千亿次($10^{11}$)基本运算，由于$2^{10}$约为$10^3$，而$2^{260}=(2^{10})^{26}\approx 10^{3\cdot 26}=10^{78}$，故所需的时间是$10^{78}\div 10^{11}=10^{67}$秒，这相当于$3.17\times 10^{59}$年(一年365天，一天24小时)。

5 需进一步讨论的问题

从以上分析来看，有理由相信，能把$TGL(8,8)$方案作为公钥体制下的一种加密方法，加密矩阵A公开，而置换矩阵P和n个2阶可逆矩阵作为解密密钥。进一步，为加大这种可能性，在$TGL(n,p)$中可取大些的n和p，这样，至少在理论上可以把$TGL(n,p)$方案作为公钥体制下的一种加密方法。限于篇幅，本文不再讨论。

小结 本文利用矩阵张量积理论，讨论了$TGL(n,p)$方案作为私钥体制下的一种数据加密方法的可行性及先进性，并详细分析了它的一种具体实用模型$TGL(8,8)$。进一步指出了把$TGL(n,p)$方案作为公钥体制下的一种数据加密方法的可能性。对于实用模型$TGL(8,8)$，在笔者近年来进行的软件研制开发工作中得到了具体的应用，取得了很好的效果。

参 考 文 献

1 Meyer C H, Matyas S M. Cryptograhy: A New Dimension in Computer Data Security—A Guide for the design and Implementation of Secure Systems. John Wiley & Sons, Inc, 1982
2 Gabidulin E M, Paramonov A V, Tretjakov. Ideals over a non-commutative ringand their application in cryptology. In: Lecture Notes in Computer Science 547, Springer-verlag, 1991. 482～489
3 Gibson J K. Severely denting the Gabidulin version of the McEliece public key cryptosystem. Designs, Codes and Cryptography, 1995, 6: 37～45
4 杜伟章，王新梅. 基于最大秩距离码的私钥加密方案. 计算机学报，2001，24(6)：650～653
5 王伯英. 多重线性代数基础. 北京：北京师范大学出版社，1985，5
6 McCoy N H. 著，刘绍谋译. 环与理想. 1983，5
7 郑雪雪. 数据安全与软件加密技术. 北京：人民邮电出版社，1995，6
8 李新晖，陈梅兰. 信息安全技术的研究发展与应用. 计算机与现代化，2000(4)：28～33