嵌入式软件可靠性测试系统及其通信研究

任洪波
（浙江大学计算机学院　杭州 310027）

摘　要　传统的嵌入式软件测试系统，不论是单机结构还是分布式结构，在长时间大规模的嵌入式软件可靠性测试时，都在数据传输方面存在相应问题。本文针对嵌入式软件可靠性测试对数据通信的实时性、延迟确定性要求，结合在国防、通信、航天、工业控制等领域广泛应用的交换式以太网技术，介绍了一种星型拓扑嵌入式软件测试系统原型。

关键词　嵌入式，软件可靠性测试，星型拓扑，交换式以太网，IEEE802.1p，EDF

Communication Research and Reliability Testion for Emedded Software System

REN Hong-Bo
(Computer Science Institute, Zhejiang University, Hangzhou 310027)

Abstract　The traditiioal testion system for embedded software system, either a single computer structure or distributed structure, has some problems in data communication while it rtests the reliability of an embedded software system for a long period of time and/or a large scale. This paper introduces a new testing prototyupe for embedded software systems with atar topology structure. The prototype is researched based on the properties of real time communication, delay addurance for embedded software sysyem and exchange Ethernet, which is widely adopted in the areas of national defense, communication, aviation.

Keywords　Embedded, Software reliability testing, Star topology, Exchange Etherner, IEEE802.1p, EDF

1　引言

随着计算机和信息处理的广泛应用，计算机系统的可靠性问题越来越得到人们的关注。而软件体系规模的日益增大及其复杂性的日益增加，使软件的可靠性问题更为突出。软件可靠性测试是在软件生存周期的系统测试阶段提高软件可靠性水平的有效途径。随着嵌入式设计成为工业现代化、智能化的必经之路，嵌入式产品深入到各行各业，为了保证系统的稳定性，嵌入式软件的可靠性测试成为嵌入式开发的一个重要环节[1]。软件可靠性测试是指为了保证和验证软件的可靠性要求而对软件进行的测试。软件可靠性测试的方法从概念上讲是一种面向需求，面向使用的黑盒测试方法。为了暴露软件在使用过程的缺陷，软件可靠性测试需要运行大量的测试用例。软件可靠性测试是长时间大规模的测试，对测试数据收发的实时性与确定性有很高的要求。

目前针对嵌入式软件可靠性测试可采用单机结构与分布式结构。单机结构不单从处理能力、数据传输方面，还是功能的扩展方面都无法满足嵌入式软件可靠性测试的要求。分布式结构将测试系统的任务、功能根据不同实时性要求分布在不同主机上，提高了系统的处理能力并便于系统功能的扩展[2]。目前更多的分布式嵌入式软件测试系统是建立在以太网上，由于以太网固有的随机性，使网络延迟的确定性较差，同时 TCP/IP 协议本身不支持实时通信，很难满足可靠性测试中的实时通信要求。

本文借用在工业以太网上应用广泛的交换技术，提出了一种针对可靠性测试的星型嵌入式软件可靠性测试系统原型。

2　星型嵌入式软件可靠性测试系统

2.1　系统组成

本系统由宿主机、数据库服务器、交换机、实时处理机、目标机组成。宿主机、数据库服务器、实时处理机之间采用星型拓扑结构，由交换式以太网连接。实时处理机与目标机可用串口或以太网连接。整个测试系统的体系结构如图 1。

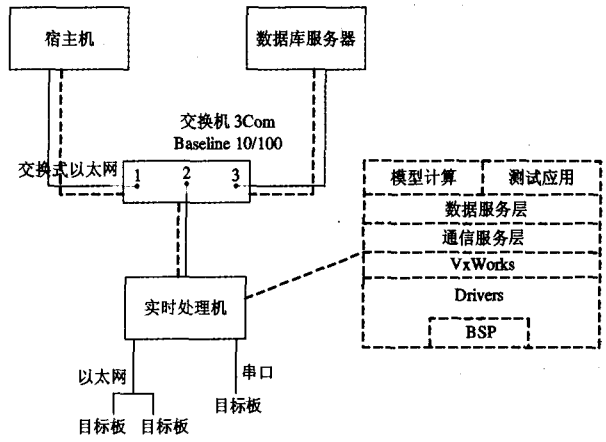

图 1　系统体系结构图

2.1.1　宿主机

宿主机(Host)通常是一台 UNIX 工作站或者 Windows NT PC 机，其主要任务是：用户命令接口，系统配置，测试用例及测试方案生成，测试脚本编写，测试过程监控，测试回放，测试结果分析和处理，可靠性评估，测试文档辅助生成。宿主机作为用户节点，应便于用户使用，因此，本系统采用 Windows 操作系统，开发面向软件可靠性测试的应用软件。同时，宿主机作为交换式以太网中的终端节点，需要对其 TCP/IP 协议进行修改来达到系统的实时性要求。

2.1.2　数据库服务器

数据库服务器用来存储软件运行及测试过程中产生的数据，捕捉瞬态故障，供事后查询和回放，并为人工智能诊断提

任洪波　硕士，讲师，研究方向：嵌入式系统软件与硬件，智能交通系统 ITS。

供数据来源。

2.1.3 实时处理机

作为测试系统的核心部件，实时处理机一般采用具备实时处理能力的工作站或者微机，它们可以是运行实时操作系统的工作站和微机，也可以是对普通操作系统进行实时扩展的微机。作为测试服务器，实时处理机负责与目标机进行交互。用户节点不与目标机直接交互信息，所有的信息都必须经过实时处理机。同样，目标机通过目标代理与实时处理机中的测试服务器交互信息。实时处理机的任务主要包括：目标系统配置；解释测试脚本，对数据进行仿真处理；生成激励信号，驱动被测试软件运行；接受测试数据，进行实时比较；测试数据实时显示。

本文采用的实时操作系统是 VxWorks。VxWorks 是高性能、可裁剪的实时操作系统。它支持广泛的兼容平台，有高性能可裁剪实时微内核 wind，有丰富的接口资源和大量的第三方产品，并且 VxWorks 有一个非常优秀的开发环境 tornado。VxWorks 应用领域包括数据网络、多媒体、工业、交通运输、远程通信、航天、安全、计算机外围设备等方面。实时处理机针对不同目标系统，通过配置文件，可以采用串口或以太网通信。

2.1.4 交换机

本文对交换式以太网的要求如下：足够的带宽；网络传输延迟确定性好；能对不同类型信息流实施不同优先级服务。华为 3Com 公司的 Baseline10/100 交换机均支持通信量优先机队列排序(IEEE802.1p)，能让网络更加有效地运行实时应用。它的每个端口支持 2 个队列，我们将其分为一个实时帧队列和非实时帧队列。通过对 IEEE802.1p 标准帧改造进而支持我们的实时通信要求。

2.2 节点互连

分布式嵌入式软件测试系统中的节点互连可以有总线型、CrossBar、星型、环形连接等多种[3]。常用的分布式测试系统通常选用基于总线的互连结构。这种无中心拓扑结构的优点是互连结构实现简单，扩展方便，网络抗毁性好。但是，当网络中节点数据较多，信道竞争成为影响网络性能的要害。特别是在测试数据容量很大的可靠性测试中，由于数据的反复拷贝，导致网络带宽严重下降，数据传输率降低。

本测试系统采用带中心点的星型拓扑结构。由于带中心点的星型拓扑结构中，网络节点的布局受到限制较小，而且当网络负荷较重时，网络的吞吐量以及网络时延性能的恶化并不剧烈，非常适合长时间大规模的可靠性测试。同时，为了达到可靠性测试的实时要求，本系统借助实时交换式以太网技术来实现星型拓扑结构。

2.3 测试流程

软件可靠性测试是为了达到或验证用户对软件的可靠性要求而对软件进行的测试；通过测试发现并纠正软件中的缺陷，提高其可靠性水平，并验证它是否达到了用户的可靠性要求。软件可靠性测试的主要步骤包括：1)模拟软件的真实运行环境；2)生成测试用例，对软件进行测试，并详细记录测试结果；3)测试结果作为软件可靠性评价模型所需的数据，通过模型计算可靠性指标；4)通过判断可靠性指标是否“达标”来决定可靠性测试是否可以停止，如果没有“达标”，则继续测试，重复上述 2～4 步，直到软件可靠性“达标”。软件可靠性测试的一般流程如图 2。

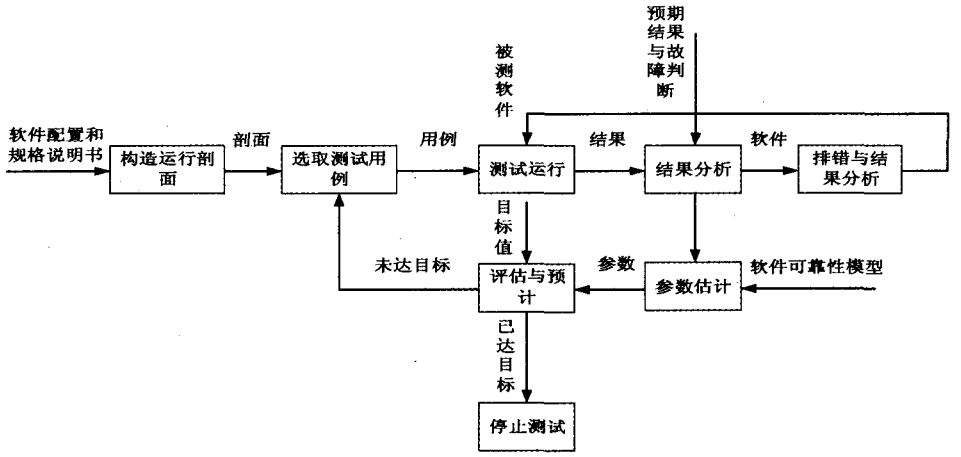

图 2 软件可靠性测试流程

由图 2 可知，测试结果的高效、准确传输是保证整个测试顺利、正确进行的关键。在嵌入式软件测试中，结果分析与测试运行是分别在宿主机与目标机中进行。两者之间的数据通信能力就显得尤为重要。

3 数据通信

本测试系统的数据通信主要包括实时处理与目标板之间的数据通信，用户节点间的通信(宿主机与数据库服务器)和用户节点与实时节点间的通信(宿主机与实时处理机)。其中用户节点与实时节点间的通信是嵌入式软件可靠性测试的重点，也是网络延迟与其不确定性对系统影响最大的部分。本文以下内容将着重介绍用户节点与实时节点间的通信问题与解决方案。

3.1 网络延迟与不确定性分析

用户节点与实时节点之间的数据传输包括实时帧与非实时帧。传统上，通过一个端口的流量必须在只有一个输出队列的缓存中保存，不论它的优先级是多大，也必须按照先进先出的方式处理。当队列满的时候，任何超出的部分都将被丢弃。此外，当队列变长时，延时也增加了。我们将所有延迟分为两类：A)基于帧长度、MAC 输出速率产生的延迟，协议处理时间和传输延迟；B)将流量的排队时间和发送时间称为：缓冲响应时间。而 B 类延迟正是网络延迟的主要部分，也是延迟不确定的主要原因。一个帧的缓冲响应时间为：

$$d_{response}=\frac{l_i+\sum_{w=1}^{i-1}l_w}{c}$$

其中 $l_i=m_i+\mu, l_w=m_w+\mu, m_i$：当前帧长度，$\mu$：帧间隔，$m_w$：排队帧长度，$c$：MAC 输出率。传统以太网不具有优先级队列特征，实时帧与非实时帧都在一个队列里。由于队列中的非

实时帧有突发行的特点，而且其长度较大，因此正是因为 w 的不确定性，从而导致了 $d_{response}$ 的不确定。为此我们采用 2 级优先级队列方式(IEEE802.1p)减少非实时帧对实时帧的影响，同时在实时帧队列中采用 EDF 算法进行调度，提高网络实时性。

3.2 节点间通信

嵌入式软件测试系统作为实时分布式系统，对系统中通信的预知性与确定性是可靠性测试系统通信成功的关键。为了使用 IEEE802.1p 排队特征，需要采用 IEEE802.1Q 格式(如图 3)。在原以太网 MAC 格式的源地址域与长度/类型域中插入一个 4 字节长的新标签域。新标签域的头 2 字节始终被设定为 0x8100，称为 802.1Q 标签类型；剩下的 2 字节中，前 3 位为用户优先级，后面 1 位是 CTI(规范格式指示器)，最后 12 位定义为 VLAN 标识符。IEEE802.1p 最多可定义 8 个优先级，我们使用其中 2 个。实时任务的优先级，非实时任务的优先级分别被设为 7 和 5。

由于非实时帧可能在交换机与源节点的 MAC 层阻塞实时帧的输出，因此 IEEE802.1p 优先级队列被引入交换机和源节点的 MAC 层。在源节点的 MAC 层，到达帧在排队前首先被封装成 IEEE802.1Q 以太网格式。根据该帧的类型来设置优先级域。交换机通过信息分类技术将实时帧与非实时帧放入 2 个队列进行输出调度。这样，不管是在源节点还是交换机处，都能保证实时帧先于非实时帧输出。

同样地，考虑到不同的实时帧有不同的实时要求，比如脚本的实时解释、实时激励的周期就要小于数据的实时显示，因此，我们在实时帧中插入 Deadline 域(如图 3)。该任务可由源节点的实时 API 完成。然后在实时队列中使用 EDF(Earliest Deadline First)调度算法，保证最终期限最早的任务更快得到服务。改造后的通信协议层次结构如图 4。

最早期限优先(Earliest Deadline First，EDF)算法是 C. L. Liu 和 J. W. Layland 在 1973 年提出的一种经典的实时 I/O 调度算法[6]。EDF 算法基于任务的截止期给任务赋优先级(离截止期越近的任务被赋予更高的优先级)，每当检测到一事件时，调度程序就将其加入等待队列中。这个等待队列根据这些任务的时限排序，最近的时限在最前面(对周期性任务时限就是它下一次发生时间)。然后，调度程序就从队列中选择第一个任务调度，就是距它最后期限最近的一个。

结束语 本文通过对交换式以太网的实时(软件)改进，提出了一种解决嵌入式软件可靠性测试中数据网络传输延迟与不确定问题的方法，并在其之上设计了一种星型嵌入式软件可靠性测试系统。采用星型拓扑结构，网络节点的布局受到限制较小，便于在系统中加入其他用户节点、故障注入系统等，同时便于将嵌入式软件可靠性测试系统集成在国防、通信、航天、工业控制等领域。下一步工作为改进星型拓扑结构的网络抗毁性差的缺点，同时改进 EDF 算法在各个节点控制其传输流。

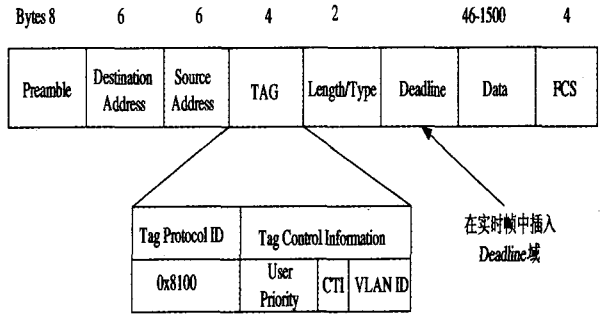

图 3 改进 IEEE802.1p 标准帧结构

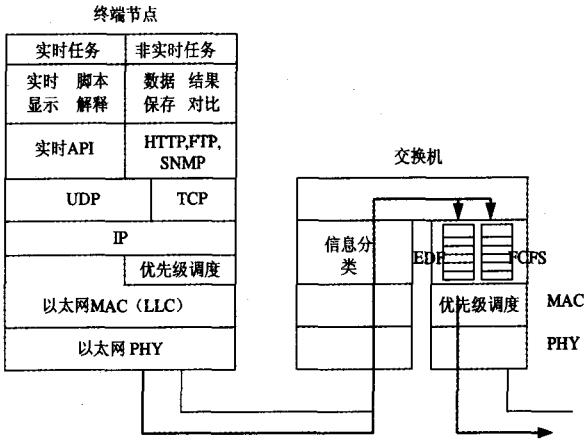

图 4 通信协议层次结构

参 考 文 献

1 Brokeman B, Notenboom E. Testing Embedded Software. 2002
2 刘斌，高小鹏，陆民燕，阮镰. 嵌入式软件可靠性仿真测试系统研究. 北京航空航天大学学报，2000，26(4)
3 崔小乐，刘斌，钟德明，等. 实时嵌入式软件仿真测试平台的体系结构设计. 测控技术，2003，22(7)
4 Zhang QiZhi, Zhang Weidong. Priority Scheduling in Switched Industrial Ethernet. American Control Conference, 2005. 8
5 Alimujiang, Yiming, Eisaka T. An Ethernet Protocol for Real-time Communications. In: SICE Annual Conference in Sapporo, 2004
6 Liu C L, Layland J. Scheduling algorithms for multiprogramming in a hard real-time environment. Journal of the ACM, 1973, 20(1)

(上接第 289 页)

结论 异常机制是 Java 错误处理技术的一大进步，它避免了类 C 程序那种随时随地测试程序、判断结果的瞬时风格，而采取了异常的分类和集中非局部处理的理念，使 Java 程序更清晰，更易于维护。但同时，我们仍然注意到了 Java 同时引入了检查型异常和非检查型异常，而检查型异常似乎是和我们上述力求避免的结果相违背，因为它迫使开发人员必须给出异常处理程序，这不仅会降低程序的可读性，使正常的程序被异常处理程序所纠缠，同时也可能会降低程序的可靠性。究竟是使用检查型异常还是非检查型异常，异常到底由谁来处理，仍然有许多争论，毕竟，程序规模有大有小，检查型异常在小规模程序里的优点并不足以引申到任何规模的程序里，也许我们应该记住，Java 异常处理的目的就是用更少的代码构建更大型、更可靠的程序。

参 考 文 献

1 Perry D E, et al. Current Trends in Excep tion Handling. IEEE Trans. Software Engineering, 2000, 26 (9): 817～819
2 Gosling A. The Java programming language, second edition[M]. Addison-Wesley, 2001
3 Eckels B. Thinking in Java, 3rd edition[M]. Prentice Hall PTR, 2002
4 Sinha S, Harrold M. Analysis and testing of programs with exception-handlingconstructs. IEEE Transations on Software Engineering, 2000, 26(9)
5 Horstmann C S, Cornell G. Core Java 2. Volume 1: Fundamentals