精确同步 Markov 测试帧的研究与设计*

王 欣，贺建飚
（中南大学，湖南 长沙 410083）

摘 要：在分析传统做法的基础上，提出了一种新的自适应的方法，在进行 Markov 测试时的链路延时和处理延时可以动态地根据运行环境自动调整，从而保证 Markov 测试帧的精确同步。使用该方法解决 Markov 测试帧的同步问题能很好地满足应用需求，在实际应用中已表现出了良好的效果。

关键词：网络性能数据；Markov 测试；同步

中图分类号：TP393 **文献识别码**：A

Study and design of mathematically synchronization Markov testing frame

WANG Xin, HE Jian Biao
(Central South University, Changsha 410083, China)

Abstract: Based on the analysis of traditional practices, this paper introduces a new adaptive method. Markov chain delay and delay disposal can be automatically dealt with according to the dynamic operating environment, so as to ensure the Markov test frame can be accurately synchronized. This method can be commendably meet the application requirements, and has shown good results in the practical application.

Key words: network performance data; Markov test; synchronization

Markov（马尔科夫）测试协议[1]作为 IS-97D 协议[2]的一部分，主要用来测试基站系统的物理层性能。Markov 协议，模拟实际情况下的语音业务特性，对前反向基本业务信道的数据传送质量进行评估。Markov 的前/反向测试帧是在 BS 和 MS 两侧分别按照一定规律产生的，测试帧的产生是一个伪随机过程。当 BS 与 MS 同步时，两侧产生的前/反向业务数据应该是相同的，即接收方可以在本地再现发送方的数据。接收方将接收到的测试帧与本地产生的帧进行比较，从而可以判断出接收到的数据是否正确。同时，BS 和 MS 还分别对发送和接收到的各种帧进行计数，根据计数器的值计算前/反向的帧错误率。

1 Markov 测试的关键问题

Markov 测试的过程逻辑上可以分解为两个环节：采集上报和测试环节。本节重点讨论如何在业务层面保证产生的测试帧精确同步，达到测试的目的。只有很好地解决了这个问题，才能从真正意义上实现 Markov 测试功能。

在现有的 CDMA2000 系统中，要实现 Markov 测试协议，必须在移动台和基站两侧维护两个伪随机数发生器，由于移动台和基站都使用了全球定位系统的时钟，因此在系统时间上能保持高度的同步，如果系统内部没有延时，则 Markov 测试协议的实现问题迎刃而解，但熟悉移动通信系统的人都知道，基站系统内部的数据传送都会存在一定程度的延时[3]，这给解决该问题带来了一定的难度。

具体的说，一个 CDMA 基站子系统可以划分为三大部分：移动台 MS、基站收发信机 BTS（包括射频和基带子系统）、基站控制器 BSC，这两个伪随机数发生器分别位于 MS 和 BSC 上，一个前向测试帧在 BSC 上产生后，会发送到 BTS 上，在基带子系统的 CHM 上进行调制操作后，经射频部分发送到与 MS 的空中接口，MS 使用分配的长码[4]对帧进行解调，与自己产生的一帧数据进行比较，累计得到前向误帧率；同理，对于一个反向测试帧，由 MS 生成，调制后经空中接口发送到 BTS，CHM 完成解调工作后将数据帧发送到 BSC 并统计误帧率。按照 Markov 测试协议，CHM 每 20 ms 会从 BSC（或 MS）收到一个测试帧，调制（或解调）后发送到 MS（或 BSC），由

* 基金项目：国家自然科学基金项目（60673165）

于MS和BTS使用的是空中接口（简称空口），而BTS的射频和基带在同一子系统内部，通常认为空口消息和射频基带子系统中的传输延时可以忽略不计，因此Markov测试协议实现的关键问题就是如何解决测试帧从BSC传送到CHM的过程中存在的延时和CHM调制解调数据的处理延时。

2 Markov测试问题的解决

本节首先简要介绍Markov测试模块的组成，在此基础上分析较为传统的使用经验数据方式，然后提出一种较好的解决方案——简单的自适应方法。

2.1 Markov测试模块的组成

操作维护中心OMC(Operation Maintenance Center)，一般是指各个电信设备制造商针对自己生产的电信设备开发的一套操作维护系统[5]，用于管理电信设备。对于CDMA2000系统的OMC，通常只管理属于基站子系统BSS范畴的网元，它们包括两类：基站控制器BSC（Base Station Controller）和基站收发信机BTS（Base Transmitter Station）。CDMA2000 OMC系统采用了3层复用的设计架构，自下而上依次为统一网管平台UEP层、公共应用框架CAF层和业务层。

支持Markov测试的移动台MS上的定时器每隔一定的时间间隔会产生一个测试帧，测试帧经过调制后，通过空中接口中的业务信道发到所属基站，基站的射频子系统（RFS）对信号进行放大后发送到基带子系统，基带子系统完成对收到数据的解调，恢复为原来的测试帧并通过BTS与BSC之间的链路发送到BSC侧的业务子系统进行处理，业务子系统将接收到的测试帧和本侧生成的测试帧进行比较和计数，定时将统计结果报给OMC的代理进程和WSF，WSF完成对帧的统计，并计算误帧率和其他数据，以多种方式呈现给用户。Markov测试的实现主要集中在业务子系统和OMC上，涉及的模块较多，主要包括OMC工作站功能WSF(Work Station Function)模块、OMC设备管理功能EMF(Equipment Management Function)模块、操作维护处理板OMP(Operation & Maintenance Processor)，非分组终端NPT(Non-Packet Terminal)Agent模块、CMP NPT Agent模块、CMP BSSAP进程、SDU进程。OMP单板上的NPT代理进程是OMC的重要组成部分，EMF端的消息到达网元后都需要经过OMP NPT Agent的处理或转发，这是实现该功能的关键节点之一。OMP代理将确定控制消息发送的下一个目的地，并将消息转发到业务进程所在单板。CMP单板上的NPT代理进程、OMC的组成部分、该进程和业务应用进程都位于CMP（呼叫处理板）上。CMP NPT Agent收到控制信号后可以调用业务进程完成Markov的呼叫、释放操作。BSSAP进程即基站子系统应用进程，按呼叫流程完成基站对手机的呼叫建立、呼叫释放等动作。SDU即选择分发单元，该进程位于SDU单板上，主要完成对信道板解调后的业务帧的处理工作。

2.2 Markov测试帧精确同步的实现

目前，就如何解决测试帧从BSC传送到CHM的过程中存在的延时和CHM调制解调数据的处理延时这些问题，较为传统的做法是使用经验数据，即当BSC产生测试帧的时候不是直接使用全球定位系统的时间，而是加上一个经验的延迟时间，该延迟时间即是预估的从BSC产生测试帧到射频系统发送出数据的时间差[6-7]。

这种采用经验数据估算延时的方式在一定程度上可以解决Markov测试的问题，但同时也存在两方面的不足：其一，系统的传输延时并不是一个静态的常量，而是一个动态值，它会随着环境的变化而变化，使用常量去模拟一个动态值，往往因为时间同步不准确导致Markov测试得到的误帧率指标高于真实值，达不到精确测量的效果；其二，经验数值往往有一定的适用范围和使用场景，不能做到自适应，当系统发生变化或Markov测试和真实情况出现严重偏差时，通常需要调整这个经验值。这些都会给维护人员和现场调试人员带来一定的麻烦，因此需要使用一种新的方式来保证基站子系统和移动台Markov测试帧的精确同步。

较好的解决方案是提供一种简单的自适应方法，即在进行Markov测试时链路延时和处理延时可以动态地根据运行环境自动调整，从而达到测试帧的精确同步。在Markov测试时，BSC侧的Markov测试的伪随机数发生器通常在选择器（SDU）上，其定时产生的测试帧经过CHM发送到MS，MS收到该帧后本地产生一个新的测试帧与该帧进行比较，若不相同则记作坏帧，对于反向误帧率，位于MS上的伪随机数发生器定时产生测试帧，经过CHM解调后发送到SDU进程，SDU收到该帧后本地产生一个新的测试帧与其比较，完成误帧率的统计。因此要解决不能精确同步的问题，最简单的做法是在SDU产生前反向测试帧时进行处理：当SDU产生将要发送的前向测试帧时，必须产生比当前时间要晚的数据包；而产生预测的反向测试帧时，必须产生比当前时间要早的数据包，只有这样，移动台侧和SDU上的前反向伪随机序列产生器才能真正同步起来。所以必须知道当产生前向测试帧时，必要的延长时间；而产生反向测试帧时，必要的提前时间。要实现这种方案必须提供一种预测机制，即根据当前的链路状况和信道板处理情况，预测将来的延时情况，还需要提供一套交互机制，告诉SDU产生GPS时间的测试帧。预测机制的实现主要在处理该测试帧的CHM上，CHM每收到1个来自SDU的前向测试帧，都会记录下收到该帧的GPS时间和发送到射频子系统的时间（也可以认为是MS收到该帧的时间，因为空口通信或射频子系统对业务包的处理时间可以忽略不计），并保存这两个时刻的差值：

CHMProcessTime=CHMSendToRFTime-

CHMRecFromSDUTime (1)

该差值可以认为是本次 CHM 处理该测试帧的时间，也可以认为是 CHM 处理下一个测试帧的时长（ms 级）。CHM 需要将该数据通过相关接口告诉 SDU 模块，便于正确产生测试帧。该接口包括的主要类型：CHM 当前的处理时长和信道板收到测试帧的 GPS 时刻。对于信道板处理时长字段，可以和 SDU 协商使用一个字节来标识：其中 2 bit 的步长域处理延时的步长；其余 6 bit 指出延时相对于步长的表示。CHM 处理时长的表示方式如表 1 所示。

表1 CHM处理时长的表示方式

步长域取值	时间步长/ms	表示范围/ms
00	125	±3.875
01	1.0	±31.0
10	2.5	±77.5
11	保留	保留

在 SDU 解析该字段时，可以按照表 1 的对应关系，使用下面的公式：

CHMProcessTime = Scale × Multiple (2)

该结构中还有另一个字段用于标识 CHM 收到 SDU 测试帧的 GPS 时刻，由于 SDU 知道自己产生该测试帧的 GPS 时刻，计算两个时间的差值可以得到 SDU 与 CHM 之间的链路延迟时间 LinkDelay：

LinkDelay=CHMRecFromSDUTime−SDUConstructFrameTime (3)

SDU 根据 CHMProcessTime 与 LinkDelay 的和得到上次发送测试帧的延迟时间，用于修正下一次产生业务帧的时间。对于由 SDU 生成的前向测试帧，SDU 将产生帧的全球定位系统时间与延时相加，就可得知 CHM 发送帧的时间，也就是基站收发信机射频前端处发送 20 毫秒帧的全球定位系统时间；对于产生的反向测试帧，需要将收到来自 MS 的测试帧的 GPS 时间减去该延迟，空间的延迟可以不用考虑，即可得到 MS 产生该帧的时刻。

本文创造性地提供了一种简单的自适应方法来实现 Markov 测试帧的精确同步，目前该方案在中兴通讯 3G 系统网管软件平台的实际应用中表现出了较好的效果。使用该方法解决 Markov 测试帧的同步问题能很好地满足 Markov 测试的需求。

本文针对 Markov 测试帧精确同步问题，在分析传统做法的基础上，提出了一种新的自适应的方法，在进行 Markov 测试时，动态地根据运行环境自动调整链路延时和处理延时，从而保证 Markov 测试帧的精确同步。

参考文献

[1] 3GPP2 C.S0025.Markov Service Option (MSO) for cdma2000 Spread Spectrum Systems.2000.
[2] 3GPP2 C.S0010.Recommended Minimum Performance Standards for cdma2000 Spread Spectrum Base Stations. 1999.
[3] 陈化钧.从3G外场测试看未来网管. 通信产业报，2004(12).
[4] 冉晓明.CDMA 扩频通信.数据通信，1998(3).
[5] 黄永康，傅范丰. 中兴ZXG10-OMC操作维护中心.通讯世界，1999，6(6)：62-63.
[6] ITU-T Recommendatio X700.Management Framework For Open Systems Interconnection (OSI) For CCITT Applications.1992.
[7] 陈小光，陈蔚薇，郭丽丽. 嵌入式软件运行剖面建模及测试用例生成[J].微计算机信息，2008，4(4)：243-245.

（收稿日期：2008-12-14）

(上接第 32 页)

```
#elif defined(CONFIG_ARM)
#define EI_SHIFT(x) ((x)*2)
```

其中 EI_SHIFT 可以查看到 8390.h 的定义。

也有直接访问外部的代码，所以要添加的还有：

```
#ifdef CONFIG_ARM
    regd = inb_p(ioaddr + 0x0d*2);
    outb_p(0xff, ioaddr + 0x0d*2);
#else
    regd = inb_p(ioaddr + 0x0d);
    outb_p(0xff, ioaddr + 0x0d);
```

：函数 outb_p 和 inb_p 访问外部 IO 的函数

这样就被解决了地址偏移的问题，这里采用预处理来添加自己的代码，不直接在原有的代码上修改，可以保证代码的完整性和可移植性，也较容易比较和发现问题。

主程序和 μClinux 中的系统文件放在同一个程序下，进行编译即可。为了提高执行效率，可以根据实际应用修改 μClinux 的部分常用代码，甚至剪切掉某些不必要的代码。

基于 μClinux 的网络构件的设计方案在硬件上简洁可靠；软件可维护性好，可扩展性好，有利于系统的后续开发，降低了系统设计的复杂性。随着嵌入式产品研究的深入，网络接口芯片的研究也会快速发展，使智能化产品的设计更趋向简单、标准、成熟。可以看出，嵌入式 μClinux 操作系统与网络将会得到更大的发展和更广阔的应用。

参考文献

[1] 周立功，陈明计.ARM嵌入式linux系统构建与驱动开发范例[M].北京：北京航空航天大学出版社，2006.
[2] 管耀武，杨宗德.ARM嵌入式无线通信系统开发实例精讲[M].北京：电子工业出版社，2006.
[3] 周立功.ARM嵌入式系统软件开发实例(一)[M].北京：北京航空航天大学出版社，2004.

（收稿日期：2008-12-17）