基于着色 Petri 网的 Internet 电话端系统业务冲突检测

郝 扬 古天龙

(桂林电子科技大学计算机与控制学院 桂林 541004)

摘 要 将描述端业务的 LESS 脚本转换为着色 Petri 网模型,可以为实现形式化方法检测端业务间的冲突提供基础。本文根据业务逻辑树节点的特性和 LESS 的定义,提出了通用的转化规则,实现了端业务的形式化建模,从而方便了业务的集成及业务间的离线检测。通过 CPN Tools 对建立的业务模型进行仿真并分析模型状态空间,检测出端业务之间是否存在冲突。最后,用典型的业务实例验证了所提方法在 Internet 电话端系统环境中的可行性和有效性。

关键词 着色 Petri 网,业务冲突,端系统

The Colored Petri Nets-based Approach for Detecting Feature Interactions in Internet Telephony End Systems

HAO Yang GU Tian-Long

(School of Computer and Control Engineering, Guilin University of Electronic Technology, Guilin 541004)

Abstract Using the transformation from the LESS scripts of end system services to the models of colored Petri nets, feature interaction in Internet telephony end systems is detected by formal methods. Based on the characters of decision-tree nodes and the definition of LESS, the universal rules of transformation are proposed. Then the formal models of end system services are obtained, and the feature integration and the off-line detection are realized easily. After simulating the models and analyzing the state space by CPN Tools, the feature interaction in Internet telephony end systems is detected. As a case study, some practical applications are introduced to show that the proposed approach works well in Internet telephony end systems.

Keywords Colored petri nets, Feature interaction, End systems

1 引言

Internet 电话是近几年兴起的、极具挑战性的 Internet 应用技术之一。基于会话发起协议(Session Initiation Protocol,简称 SIP)的 Internet 电话,具有简单、灵活的特性,它整合了传统的语音及增值服务,能够支持绝大多数 ITU-T(国际电信联盟电信标准化组织)的能力集 1 和能力集 2 中的业务,并可以为其他更多的增值应用服务商提供标准的具有高扩展性的平台。因此,基于 SIP 的 Internet 电话已经受到越来越多的关注并得到了迅速发展。

然而,现有的 Internet 电话网络中配置业务的方法还不够成熟,这种不成熟所带来的最严重问题是业务冲突。对业务冲突问题的研究始于 20 世纪 80 年代的贝尔实验室,当时的研究主要局限于传统电信系统的开发过程。随着技术的发展,Internet 电话带来了一些传统电信系统无法承载的更加丰富和灵活的业务。正是由于 Internet 电话系统中业务种类的增加和定制灵活性的增强,业务的创建和配置变得越来越复杂,业务冲突问题依旧突出。特别在 Internet 电话端系统中,用户可以通过端系统业务语言(Language for End System Services,简称 LESS)创建自己预设的业务。由于用户在设计自己的业务时很少从整体上考虑其它业务,往往会造成业务冲突,严重影响 Internet 电话网络的正常运营[1]。

业务冲突检测能够为业务冲突的处理和解决提供基础,它分为离线检测和在线检测两种[2]。离线检测指通过业务验证和仿真工具,以静态的方法检测出可能发生冲突的业务属性对;在线检测则指在具体执行时对业务呼叫实例进行分析。目前,形式化方法已被证明是一种非常有效的离线业务冲突检测方法。形式化方法通过形式化语言严格的语法、语义定义,能够保证描述结果的无二义性,避免在业务属性描述阶段就产生业务冲突,同时可以提供推导和验证工具,以支持对业务冲突的检测。着色 Petri 网(CPNs)作为形式化方法中的一种,特别适用于通信、同步和资源共享起主要作用的系统。根据该方法所建立的模型的执行以及所用的仿真环境能较为真实地反映出实际运行中的情况。本文采用着色 Petri 网形式化描述 LESS 脚本的语义,实现了用形式化方法对 Internet 电话端系统业务(以下简称为端业务)建模,并利用着色 Petri 网的构造、仿真和分析工具 CPN Tools 对系统整体模型进行仿真和状态空间分析,从而检测出端业务之间是否存在冲突。通过分别检测分析两个端业务的 CPNs 模型以及对两个业务合并之后的 CPNs 模型进行检测,验证了本文所提方法。

2 基于 CPN Tools 的呼叫控制过程建模

着色 Petri 网是由丹麦的 Jensen Kurt 于 1981 年在 Petri 网基础上定义的一种具有层次性的高级网系统。它既可以图形化地直观描述系统的功能结构,同时具有严格的数学定义。着色 Petri 网模型适用于仿真和分析庞大并且复杂的系统,因为可以通过构造分层模型,标记值和弧表达式可以表示复杂信息,辅之以成熟并且经由严格测试的构造、仿真和分析工具。对于一个大系统,在单一页面上进行仿真既不方便也过于庞大,但是如果引入分层机制来将网模型分解成多个页面,

郝 扬 硕士研究生,研究方向为下一代网络和形式化技术;**古天龙** 教授,博士生导师,研究方向为形式化方法、协议工程、符号模型检验、Petri 网和离散事件/混杂系统理论及应用等。

并用置换变迁来实现这种能力，则相对简单。本文所使用的建模和分析工具采用丹麦大学提供的 CPN Tools(v2.2)软件。该工具带有语法检查功能，对违反 CPNs 数学定义的 CPNs 结构和行为都可以自动做出检查。

CPN Tools 可以定义的颜色集主要包括基本颜色集和复合颜色集两大类。文中模型主要使用的基本颜色集有 User(用户)、Message(消息)、CallTime(呼叫时间)和 Level(呼叫级别)等。对于复合颜色集，主要定义以乘积颜色集为主的颜色集，乘积颜色集是两种或多种颜色集的笛卡儿积所产生的颜色集。例如，color UserxUserxMessage = product Userx-User * Message，定义了呼叫过程中所包含的主叫、被叫和消息内容。

在 CPN Tools 描述方法中，状态通常用库所(place)表示，图例中以椭圆表示；事件则用变迁(transition)表示，图例中以长方形表示。库所中若存放托肯(token)，则托肯必须属于该库所的颜色集类型。库所和变迁用有向弧连接，弧线上的表达式采用函数语言 CPN ML 编写。

要近似模拟实际业务的执行流程，需要创建大量复杂的 CPNs 图形。为了增强 CPNs 图形的可读性和业务模块的可重用性，本研究在建模过程中充分运用分层网的概念及其实现方法变迁替换(Substitution Transition)，模型将一次呼叫控制过程细分为多个子网，并在子网中描述涉及到的增值业务模块。模型建成后，运用 CPN Tools 对模型进行仿真，检测是否会发生业务冲突。

本文建模的基本思想是将 Internet 电话系统中的呼叫控制过程从业务流程角度上进行简化，提炼出引发冲突的主要通信参数，重点描述呼叫过程中信息流的流向及业务逻辑的执行步骤，从而模拟出一次呼叫过程。

2.1 模型顶层图

基于 SIP 的 Internet 电话系统中的呼叫是通过 INVITE 发出请求、成功响应和 ACK 确认请求的三次握手来实现的[3]，即当主叫用户代理要发起呼叫时，它构造一个 INVITE 消息，并发送给被叫。当 INVITE 消息经代理服务器到达被叫用户代理后，若被叫决定接受该呼叫，就回送一个成功响应。主叫方收到成功响应后，向对方发送 ACK 请求。被叫收到 ACK 请求后，呼叫成功建立。根据此呼叫过程建立的 CPNs 模型顶层图如图 1 所示。它包括了以下组件：

(1)4 个代表不同网络设备的替换变迁，分别是端系统 1(End System1)、端系统 2(End System2)、端系统 3(End System3)和代理服务器(Proxy server)。在实际应用中，可以依据网络结构变化，自行设计并添加网络设备，将其在对应子页(subpage)进行细节实现。

(2)连接网络设备的库所，存储转发呼叫信息流。

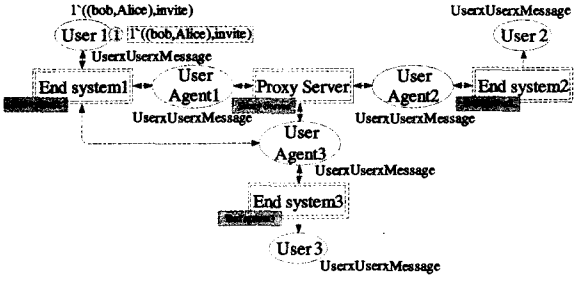

图 1 模型顶层图

在 SIP 环境中的用户使用 SIP 统一资源定位器(URL)来标识自己。SIP URL 的格式和 e-mail 地址相似，通常由一个用户名和一个域名组成，形如：SIP:bob@f-oo.com。为了简单起见，本文所建立的模型中将直接用用户名表示 SIP URL。图 1 中库所 User1 拥有一个初始托肯 1`((bob,Alice),invite)，表示用户 bob@foo.com 即将发起对用户 Alice@example.com 的一次呼叫。

2.2 替换变迁重新描述举例

替换变迁在各自的子网中重新进行描述。图 2 为顶层图中 End System1 被重新定义后的具体描述。本文中的端业务添加在被叫方的端系统中，因而主叫方的端系统只具备控制信息流及路由选择的基本功能。

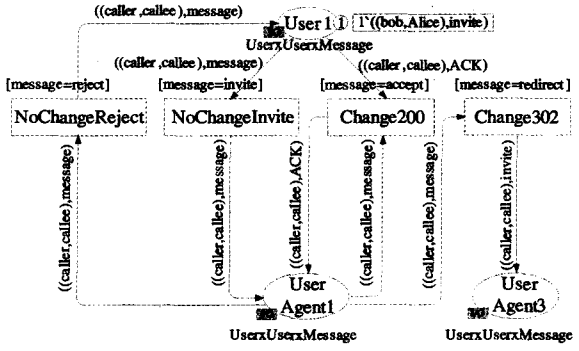

图 2 End System1 的具体描述

图 2 中，端口为 I/O 类型的库所对应于顶层图中与 End System1 的连接部分，来自其它子网的呼叫托肯存放于这些 I/O 类型的库所中。

3 端业务集成和冲突检测

端系统是指可以发起和/或接收信令消息及媒体流的设备，如普通电话、SIP 终端、PC 电话客户端等。端系统可以发起呼叫，并可以接收、拒绝或前转来电。LESS 是一种专门用于创建端业务的脚本语言，它继承了呼叫控制语言(Call Processing Language，简称 CPL)的树状结构、无循环且无递归等特性，便于进行程序验证以及文本表示和图形表示之间的转化。在此基础之上，LESS 还扩展了 CPL 语言。CPL 缺少对用户代理需要的其它行为的支持，如 call、accept、transfer 以及用户交互行为等，此外 CPL 也不能处理计时、presence 信息、即时消息和其他的网络应用等通信行为。基于以上 CPL 的各种不足，通过扩展 CPL 语言而定义了 LESS[4]。

LESS 中的呼叫决策过程包含 3 步：引入触发条件 trigger、基于 switch 将呼叫决策分别转发和执行通信行为 action。trigger 代表触发 LESS 脚本的事件，trigger 必须作为业务逻辑树的根节点；switches 用来验证事件的内容，如 address-switch 用来验证来话者的地址并决定将要执行的行为，一个 switch 最多包含两个分支，switch 必须为 trigger 或是另一个 switch 的子节点；action 能改变呼叫状态和呼叫内容，并且只有 action 能作为 LESS 业务逻辑树的叶子节点。另外，modifiers 用来设置行为的参数。当某 trigger 触发多个不同的行为时，多个 LESS 脚本之间就可能会产生冲突[1,5]。

为了实现用形式化方法检测端业务之间的冲突，需要将每个端业务进行形式化描述，即将端业务的 LESS 脚本转换成对应的 CPNs 模型。利用 XML 开发工具 Stylus Studio，可以自动生成 LESS 脚本的树状结构图，从而得到由 LESS 脚本描述的端业务所对应的业务逻辑树。根据该逻辑树的节点

特性和 LESS 的定义[6]，可以将每个端业务的 LESS 脚本转换成对应的 CPNs 模型，其主要的转换规则包括：

规则 1 根据触发条件 trigger，将端业务添加在适当的端系统中。若 trigger 为 incoming，则添加在被叫端系统中；若为 outgoing，则应添加在主叫端系统中。

规则 2 基于 switch 判断呼叫决策转发的条件。若已经在整个 CPNs 模型中给出了条件内容的定义，则将该内容参数直接作为此次呼叫转发的 guard 函数；若条件的内容在整个 CPNs 模型中还没有定义，则添加表示该条件内容的库所，并将条件中的具体参数内容作为库所的初始托肯，库所与该 switch 中包含的通信行为 action 相连。LESS 中定义了 6 种 switch，其作用、自带参数和输出参数均不同，如表 1 所示。

表 1 LESS 中定义的 6 种 switch

switch 类型	作用	自带参数		输出参数	
		参数名	参数值	参数名	CPNs 表示
address-switch	基于地址判断呼叫决策转发的条件	user	host IP	is	=
		field	origin/original-destiation/destination	contains	substring
		subfield	address-type/user/display/host/port/tel		
priority-switch	基于优先级判断呼叫决策转发的条件	——	——	less	<
				greater	>
				equal	=
string-switch	基于字符串判断呼叫决策转发的条件	user	host IP	is	=
		field	subject\user-agent\organization\display	contains	substring
language-switch	基于语言判断呼叫决策转发的条件	——	——	matches	=
status-switch	基于状态判断呼叫决策转发的条件	uri	user IP	less	<
				greater	>
				equal	=
		status-name	active-calls\presence\mood\activity	is	=
				contains	substring
time-switch	基于时间判断呼叫决策转发的条件	格式较为复杂，自带参数及输出参数较多，在实际的转换过程中将根据具体涉及的参数作出相应转换，这里省略。			

规则 3 将通信行为 action 作为新增的变迁加入到 CPNs 模型中。其中有一些 action 含有输出参数，则将输出参数的内容作为新添加库所的初始托肯，在获得一定的托肯时激发相应的 action 变迁。而那些不包含输出参数的 action 则直接作为变迁添加在 CPNs 模型中。LESS 中定义了 9 种 action，其中不包含输出参数的分别是 accept、reject、redirect、terminate、mail、log 和 wait，而包含输出参数的是：

(1)authenticate：输出参数有 success 和 failure。

(2)call：输出参数有 accepted、busy、failure、noanswer 和 redirection。

根据上面介绍的规则将每个端业务的 LESS 脚本转换成对应的 CPNs 模型，并在子网 End System2 中分别描述出来。通过 CPN Tools 分析各个模型的状态空间，验证该模型的仿真结果是否满足端业务的设计需求。再将这些模型整合，通过仿真并分析整个系统的状态空间判断这些端业务之间是否会产生冲突。本研究中以两个端业务为例来讨论。

3.1 端业务 1 的 CPNs 模型

假设 Alice@example.com 定制了端业务 1，将接听来话者为 bob@foo.com 或者优先级为“emergency”的来电，所有其它的来电将被拒绝。由于篇幅有限，这里不详细描述各个端业务的 LESS 脚本。根据工具 Stylus Studio，自动生成 LESS 脚本的树状结构图，从而得到了 LESS 脚本对应的业务逻辑树，如图 3 所示。

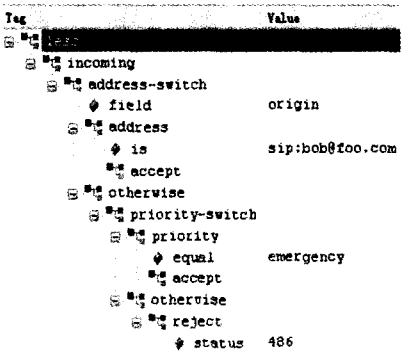

图 3 端业务 1 的业务逻辑树

脚本逻辑由一个共同的根节点(less)开始。当 LESS 脚本运行时，先执行树中的根节点，根据节点执行的结果，服务器判断节点的输出是什么，然后执行此输出指向的下一个节点，直至到达叶节点为止。

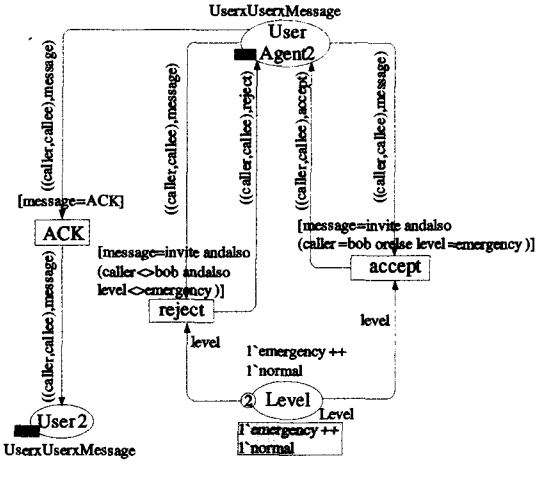

图 4 端业务 1 的 CPNs 模型

按照前面的转换方法由图 2 可知，该业务的通信行为有两个：accept 和 reject。将这两个行为作为子网 End System2

中新添加的两个变迁，把激活这两个行为的触发条件分别作为新增变迁的 guard 函数，另外还要描述信息流的内容。因此，变迁 accept 的 guard 函数在 CPN Tools 中表示为：message=invite andalso (caller=bob orelse level=emergency)，变迁 reject 的 guard 函数在 CPN Tools 中表示为：message=invite andalso (caller◇bob andalso level◇emergency)。图 4 为该业务脚本转换后的 CPNs 模型。

通过对端业务 1 的 CPNs 模型仿真和状态空间分析可知，端业务 1 能够正常运行，且仿真结果与预期的结果相符。

3.2 端业务 2 的 CPNs 模型

假设 Alice@example.com 定制了端业务 2，将所有 9：00 至 17：00(在 CPN Tools 中定义为 T1)之间的来话都转接到 specialist@foo.com，其它的来话则接听。可根据上文提到的方法将端业务 2LESS 脚本转换成 CPNs 模型，如图 5 所示。

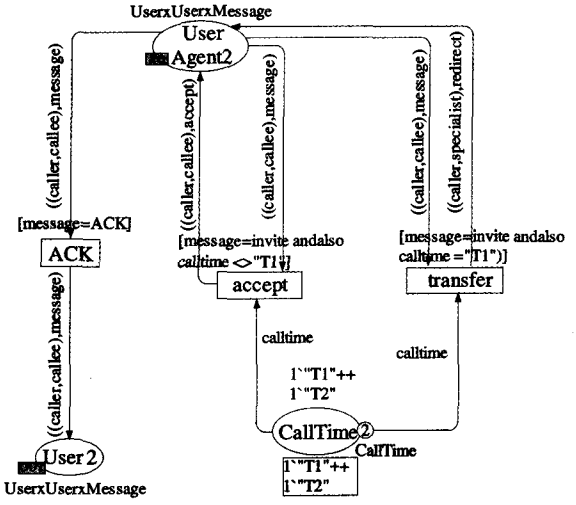

图 5 端业务 2 的 CPNs 模型

根据仿真和状态分析结果可知，端业务 2 能够正常运行。

3.3 端业务的整合

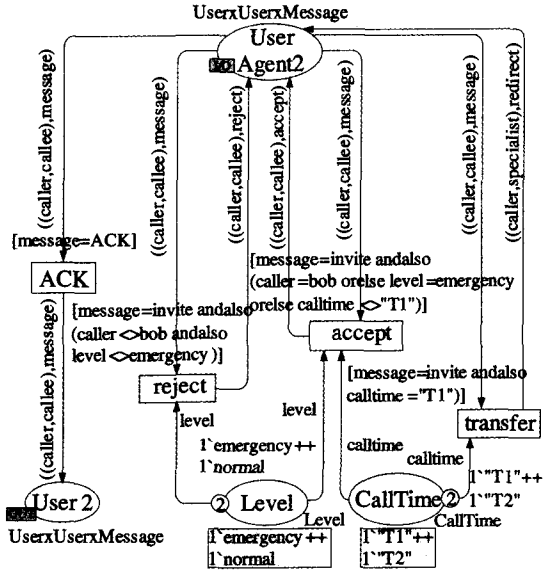

图 6 整合端业务 1 和端业务 2 之后的 CPNs 模型

设着色 Petri 网结构是一个六元组 CPN=(S,T,F,IP,OP,RC)，其中 S 表示库所集合，T 表示变迁集合，F 表示有向弧集合，IP 表示输入端口集合，OP 表示输出端口集合，RC 表示各个变迁 guard 函数的集合。端业务 1 的 CPNs 模型表示为 $A=(S_1,T_1,F_1,IP_1,OP_1,RC_1)$，端业务 2 的 CPNs 模型表示为 $B=(S_2,T_2,F_2,IP_2,OP_2,RC_2)$，则将这两个模型合并在一起的 CPNs 模型为 $A*B=(S,T,F,IP,OP,RC)$。此时，$S=S_1\cup S_2$，$T=T_1\cup T_2$，$F=F_1\cup F_2$，$IP=IP_1\cup IP_2$，$OP=OP_1\cup OP_2$，$RC=RC_1\cup RC_2$。以变迁 accept 的 guard 函数为例，它在端业务 1 模型中表示为 RC_1：message=invite andalso (caller=bob orelse level=emergency)，在端业务 2 模型中表示为 RC_2：message=invite andalso calltime◇"T1"，则在合并之后的模型中表示为 RC：message=invite andalso (caller=bob orelse level=emergency orelse calltime◇"T1")。整合端业务 1 和端业务 2 之后的 CPNs 模型如图 6 所示。

3.4 冲突检测

CPNs 的状态空间图(OccurrenceGraph，简称 OG)分析方法的基本思想是构造一个图形，其间包含对应于每一个可达状态标识的节点和对应于每一个事件绑定元素的弧。状态空间报告中包含有许多 CPNs 模型的行为属性信息，通过对其进行研究，可以发现系统是否像预期的那样工作。因此，可以借助于 CPN Tools 所包含的状态空间自动生成工具来生成状态空间分析报告，以便提供着色 Petri 网行为属性的一些关键信息。

采用的呼叫实例是 1`((bob,Alice),invite)，表示用户 bob@foo.com 即将发起对用户 Alice@example.com 的一次呼叫，同时发出 invite 请求。根据状态空间分析报告可知，仿真过程结束后将产生 5 个终止标识，分别是 24、25、26、27 和 31。终止标识表示的状态是不再有使能变迁存在的状态，即系统的最终状态。通过 ML 标准函数 print(NodeDescriptor)能够查询这些终止标识的内容，例如执行 print(NodeDescriptor 31)得到的结果如图 7 所示。

```
print(NodeDescriptor 31)
31:
End_System1'User_Agent3 1: empty
End_System1'User_Agent1 1: empty
End_System1'User1 1: 1`((bob,specialist),ACK)
Proxy_Server'User_Agent3 1: empty
Proxy_Server'User_Agent2 1: empty
Proxy_Server'User_Agent1 1: empty
End_System2'CallTime 1: 1`"T2"
End_System2'User_Agent2 1: empty
End_System2'User2 1: empty
End_System2'Level 1: 1`emergency++
1`normal
End_System3'User3 1: 1`((bob,specialist),ACK)
End_System3'User_Agent3 1: empty
top'User1 1: 1`((bob,specialist),ACK)
top'User_Agent1 1: empty
top'User_Agent2 1: empty
top'User2 1: empty
top'User3 1: 1`((bob,specialist),ACK)
top'User_Agent3 1: empty
val it = () : unit
```

图 7 终止标识 31 的内容

比较所有终止标识的内容之后发现，终止标识 24、25、26、27 所表示的最终状态都建立了用户 1 和用户 2 之间的通话，即库所 User1 和 User2 的最终托肯都为 1`((bob,Alice),ACK)，只是各自的使能条件不同。而终止标识 31 反应的系统最终状态是建立了用户 1 和用户 3 的通话，即库所 User1 和 User3 的最终托肯为 1`((bob,specialist),ACK)，此时的使能条件为 caller=a 并且 calltime="T1"，但是在此使能条件之下，系统也能够到达终止标识 24 和 26。由此可见，端业务 1 和端业务 2 之间产生了冲突，即当主叫方 bob@foo.com 在时间 9:00～17:00 之间发出对定制了端业务 1 和端业务 2 的被叫方 Alice@example 的呼叫时，系统不能正常运行。

因此，通过本文所提方法对端业务建模，可在系统分析阶段检查出端业务间的潜在冲突，尽早修改业务设计方案，减小业务实现后的维修成本。

结论 本文提出了端业务 LESS 脚本到 CPNs 模型的通用转化规则，实现了用形式化方法对 Internet 电话端系统业务建模分析。利用 CPN Tools 对所建模型仿真并分析系统状态空间，最终检测出业务冲突。目前已经研究了该方法在信令服务器和端系统中的应用，接下来将把这两种应用结合在一起，研究信令服务器中的 CPL 脚本和端系统中的 LESS 脚本之间的业务冲突。

参 考 文 献

1 Wu Xiaotao, Schulzrinne H. Feature interaction in Internet telephony end systems:[Technical Report]. New York: Columbia University, 2004
2 Keck D O, Kuehn P J. The Feature and Service Interaction Problem in Telecommunications Systems: a Survey. IEEE Transactions on Software Engineering, 1998, 24(10): 779～796
3 张智江，张云勇，刘韵洁. SIP 协议及其应用. 北京：电子工业出版社，2005. 178～185
4 Wu X, Schulzrinne H. Handling feature interactions in the language for end system services. In: Reiff-Marganiec S, Ryan M D. eds. Feature interactions in telecommunications and software systems VIII, Amsterdam: IOS Press, 2005. 270～287
5 Wu Xiaotao, Schulzrinne H. End system service examples: [Technical Report]. New York: Columbia University, 2004
6 Wu X, Schulzrinne H. Less: Language for end system services in internet telephony. Internet Draft I-D, Internet Engineering Task Force, February 2005
7 Jensen K. An introduction to the practical use of coloured Petri nets. Lecture Notes in Computer Science, 1998, 1492:237～292
8 Christensen S, Mortensen K H. Design/CPN ASK-CTL Manual, version 0.9. 1996
9 Jensen K, Christensen S, Kristensen L M. CPN tools occurrence graph manual, version 0.1. 2002
10 Xu Yiqun. Detecting feature interactions and feature inconsistencies in CPL:[Master thesis]. School of Computer Science, University of Ottawa, 2003

（上接第 25 页）

由于 PNC 之间无法直接监听彼此的消息，则它们并不知道自己的 Beacon 是否应该滞后以及偏移量是多少。一种方法是依靠两个 PNC 之间的 DEV 节点对其所属 Piconet 的 Beacon 信息进行重传，并通知到其周围节点。在这里，DEV 起到了中介或者转发器的作用，这种 Beacon 的重传称为 HB (Heartbeat)。两个互相未知的 PNC 通过 HB 达到同步的过程如下(A 表示原 Piconet PNC，B 表示后续进入的干扰 Piconet PNC)：

(1)B 请求信息后，收到从 DEV 发送的 HB，得知 A 的存在，发送自己的 Beacon 并在其中标记 A 为 SEEN 状态；

(2)DEV 收到 B 的 Beacon，看到 A 的状态已被标记为 SEEN，则标记 B 为 IDENTIFIED；

(3)DEV 发送 HB；

(4)A，B 听到 HB 之后，决定二者主次或先后关系，次级 PNC 将 Beacon 进行相应偏移，标记整个 CTAP，并发送 CTA 请求给主 PNC。DEV 收到该 Beacon，将 CTA 请求复制到 HB 中；

(5)主 PNC 收到 HB，分配 CTA，相应调整 Superframe 长度，然后发送 Beacon；

(6)DEV 收到 Beacon，复制 CTA 信息到 HB，然后发送 HB；

(7)次级 PNC 收到 HB，完成最终同步和偏移。

上述消息传递过程比较清晰，整个同步过程需要 6 个 Superframe(步骤 5、6 针对同一 Superframe)参与。另外给出一种替代方案，即将 Superframe 的一小部分预留给 PNC 作为 Beacon 时隙分配。例如，将设好的预留区分为 4 个 Beacon slots。在图 7 中，当 PNC 4 作为新节点进入时，通过向 DEV 请求信息得知预留区中已经有两个时隙分别被 PNC 1 和 4、5 使用，则它可以申请第三个 Beacon slot 并且更新自己的 Beacon 信息，整个同步过程可以在一个 Superframe 中完成。方案 1 灵活性强，但建立过程复杂；方案 2 实现简单，但可扩展性一般。二者各有优劣，实际应用中可结合使用。

3.3 路由、调度及其它问题

由于 UWB 中 Piconet 接入点在时间和放置地点具有不规律性，因此设备在 Inter-piconet 网内的通信非常复杂。如何解决其路由问题及 PMP 节点调度问题，以保证网间通信顺畅成了目前研究的热点。

在组网和路由方面，困扰的主要问题是路径的选择。目前的主要路径搜索模式为地址表路径搜索模式和需求搜索模式。这两种方式要么需要通信节点具有庞大的记忆体，要么在网内发送广播信息，容易造成路径确认延迟或网内泛滥的询问路径信息，而一种基于 Piconet 设备地址的自定义路径方式相对较为适合 UWB 的随机组网模式。它根据设备地址大小的不同，确认每个节点的网内通信范围。当两个节点需要发送信息时，发送节点会判定目的节点是否在自己的通信范围之内。如果没有，则上传信息给主节点，由主节点传给相应的分节点，直到找到目的节点。这种方法基本不会增加节点记忆体开销，并且路径方式是唯一确定的，结构清晰，搜索路径快捷。

而在 PMP 节点调度方面，由于作为 PMP 的 DEV 单元以时分方式存在于不同的微微网中，PMP 节点的轮询原则上可采用循环轮询、优先级轮询等策略。但是鉴于不同 piconet 中设备接入的随机性，拟采用蓝牙中跳模式调度方式来解决此问题。

跳模式的本质是跳节点在所有与其相关的微微网中使用相同的约会窗口序列，并结合简单的信令协议，使所有跳节点的对端都知道跳节点是否在每个约会窗口存在于特定的链路中，从而减小时延。在跳模式中，当主设备在约会点开始轮询时，跳从设备向主设备通知其存在。此外，跳从设备可以为每个微微网预先分配一定数量的约会窗口，以加强与微微网内部调度策略的联系，同时这种跳模式可以允许新的微微网或节点设备加入，较好地满足了其接入随机性。

结束语 本文针对目前亟待解决的 Inter-piconet 网间的无干扰交互与通信问题，提出了建立完善的 Piconet 网间通信流程的解决方案，通过使用 LAS 正交码等方法分析解决了不同 Piconet 的码字分配干扰问题；提出逻辑微微网模型，通过 Beacon 偏移来实现 Piconet 之间的同步机制；并对路由及调度等问题逐一探讨，完成建立连接、同步、码字分配、路由等一系列过程，使得彼此未经协调的 UWB Piconet 之间能够并存运作。下一步我们准备就 Piconet 同步机制进行仿真，并且针对 UWB 组网的特点对其路由策略做进一步优化，从而促进 inter-piconet 网间通信问题的标准化。

参 考 文 献

1 IEEE 802.15.3. Wireless MAC and PHY Specifications for High Rate WPANs [S], 2003
2 徐振阳，窦文华. 无线传感反应网络综述[J]. 计算机科学，2005 (09)
3 MBOA Wireless MAC Specification for High Rate WPANs [S]. 2004
4 DS-UWB Physical Layer Submission to 802.15 Task Group 3a [S]. 2004
5 Chol B-J, Hanzo L. On the design of LAS spreading codes. In: Proc. 56th IEEE Vehicular Technology Conference [C]. vol 4. Vancouver, Canada, September 2002. 2172～2176
6 MDP Beacon Alignment [EB/OL]. http://www.meshdynamics.com/WPAN.html, 2004