FACT 协议研究

李 红 刘卫东 林 闯
(清华大学计算机科学与技术系 北京100084)

摘 要 FACT 协议是一种网络设备中用来进行控制元素和数据转发元素分隔的协议。控制元素通过 FACT 协议在分布式的环境中以 master/slave 方式控制转发元素。本文先简单介绍了一下 FACT 协议,然后就这个协议对网络设备的可扩展性、可延展性、高可靠性支持进行了分析,并给出了它在基于 NP 的路由器中的应用和作者对它的原型系统实现。

关键词 FACT 协议, ForCES, 基于 NP 的路由器

Research on FACT Protocol

LI Hong LIU Wei-Dong LIN Chuang
(Department of Computer Science and Technology, Tsinghua University, Beijing 100084)

Abstract FACT protocol is a protocol defined for forwarding element and control element separation (ForCES[1]). Control element[2] could direct forwarding element[2] of a network element[2] in master/slave architecture in a distributed environment. The article firsts introduces FACT protocol and investigates it's extensibility, scalability and high availability support for network element, presents its application in a NP based router and the author's design of FACT's prototype implementation.

Keywords FACT protocol, ForCES, NP based router

1 引言

随着快速转发网络设备的发展,如网络处理器、ASIC 等,以及信令,路由控制协议和许多第三方控制层软件的发展,对这些分开的逻辑功能,如何制定一个标准使它们对外界看起来成为一个功能整体的问题越来越突出了。

FACT 协议是一个用来在 IETF 定义的 ForCES 结构框架[1,3,5]中为控制层和数据转发层的逻辑分开的功能间进行信息交互的协议。它是一种 Master/Slave[1,3]结构的协议。控制元素(Control element)以 Master/Slave 的模式控制数据转发元素(Forwarding element)(数据转发元素是 slave, 控制元素是 master)。本文给出了作者对 FACT 协议的研究和分析,以及它对网络设备可扩展性和可用性方面的影响,并给出了作者对它的原型实现。

2 FACT 协议

FACT 协议是一种工作在分布式环境中 master/slave 结构的协议。CE(master) FACT 协议为很多控制层的协议和软件提供服务,比如 OSPF, SNMP, RSVP 协议等。FE(Slave) FACT 协议接受 CE 的控制,并且为控制层用户转发分组,向 CE 报告异常事件和统计数据。

2.1 FACT 协议的基本概念

控制元素 CE(Control Element): 是实现 ForCES 协议(FACT 协议)的逻辑实体,它可以指示1个或多个转发元素如何处理分组。控制元素有执行控制和信令协议的功能。

转发元素 FE(Forwarding Element): 是实现 ForCES 协议(FACT 协议)的逻辑实体,转发元素可以通过 ForCES 协议按照控制元素的指示利用下层的硬件为每个分组提供处理。

ForCES 网络元素 NE(ForCES Network Element): 包括一个或多个控制元素和一个或多个转发元素。对于网络元素外的实体来说,网络元素代表一个点。

转发元素模型(FE model): 描述1个转发元素的逻辑的处理功能的模型。

ForCES 协议元素 PE(ForCES Protocol Element): 执行 ForCES 协议的 FE 或 CE。

FE 模型(FE model)[4]: 描述 FE 的逻辑处理功能的一种模型。

逻辑功能块 LFB(Logical Functional Block): 数据路径中定义的细粒度的逻辑上分开的分组处理操作。LFB 可以认为是组成 FE 模型的基本模块。

2.2 FACT 协议包格式

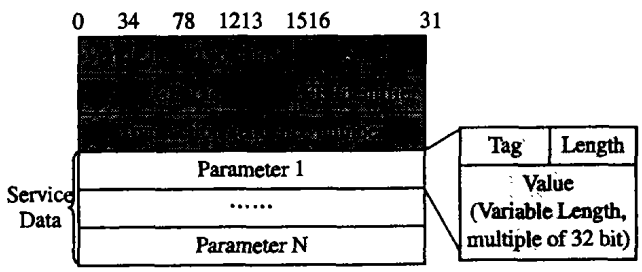

图1 FACT 消息格式

如图1所示,FACT 协议数据包由包头和服务数据两部分组成。FACT 包头包括版本、消息类别、消息类型、消息优先级、消息长度(包括包头和服务数据部分)、控制元素 CE 和数据元素 FE 的编号、消息的传输序列号。服务数据包括若干(可以为0)个以 TLV 格式封装的变长参数。

*)本文由国家自然科学基金900104002和 Intel 公司支持。李 红 硕士研究生,主要研究方向:计算机网络,网络协议,QoS 控制;刘卫东 副教授,主要研究方向:计算机网络,分布式信息系统,QoS 控制;林 闯 教授,博士生导师,主要研究方向:计算机网络,性能评价,QoS 控制。

目前,FACT 协议已经定义了6类消息:PE 联结类、能力控制类、状态维护类、流量维护类、事件报告类以及厂商自定义的类,这几种消息类型体现了 FACT 协议的主要功能:为控制层和数据转发层建立联接,支持 CE 对 FE 功能的控制和配置,维护整个网元的状态,保证数据转发层和控制层之间的业务流能够正常地流动,数据转发层的故障管理。

当一个转发到控制层的分组没有携带它的 IP 头字段时,它的优先级通过 FACT 消息显示出来。协议定义中的优先级字段是为了支持大业务流量时哪种服务的业务流应当获得更高的优先级,哪种可以被丢弃。例如,OSPF 报文应当比 ping 报文有更高的优先级,主动网络中的代码下载应当比普通的控制协议的分组的优先级高。

2.3 FACT 协议的应用环境

一个网元 NE 通常包括不止一个 CE 和 FE。FACT 协议的 master 模块执行在 CE 中,slave 模块运行在 FE 中,如图2所示。一个或者多个 CE 可以组成一个 CE 组,组内的各个 CE 彼此合作,但它们的地位并不是均等的。主 CE 将控制和配置 FE,而备份的 CE 和 FE 是用来提供 fail-over[3]支持,从而提高网元的可用性。

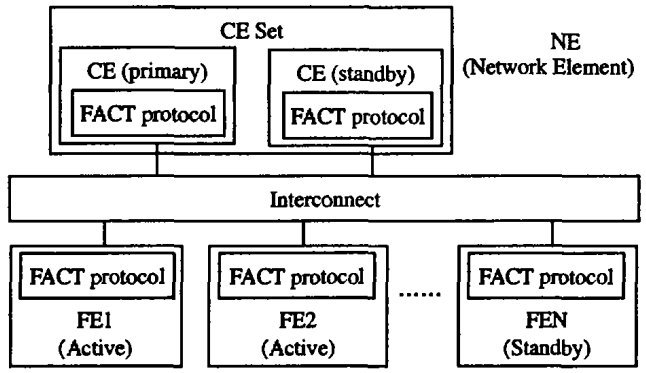

图2 FACT 协议的应用环境

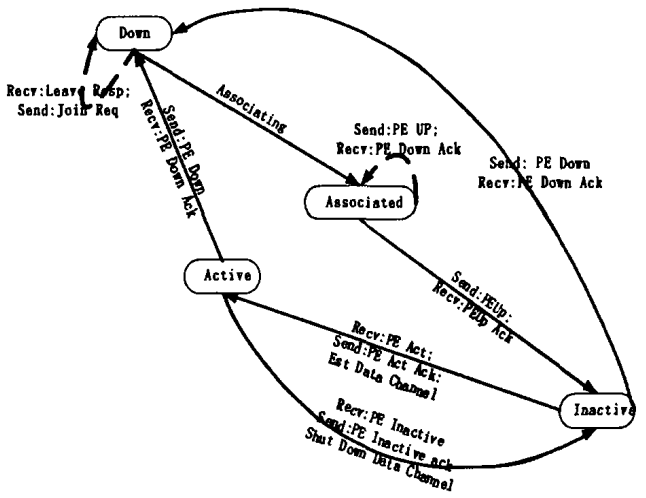

图3 PE 的状态转换图

2.4 协议的工作流程和 PE 的状态转换图

FACT 协议的执行被分成2个阶段:联接阶段和联接后阶段。在进入联接阶段之前,每个 CE 和 FE 都已经知道自己将要控制哪些 FE 和要被哪些 CE 控制,这些由联接前协议配置。

FACT 协议的工作流程如下:在联接建立阶段,FE 向 CE 发送"join request"消息,然后如果 CE 可以接受这个 FE 的请求,就发送"join response"给 FE,否则就发送"leave response"给 FE,FE 将继续向其他的 CE 发送联接请求,如果全部失败,将过指定的时间之后再重发。FE 成功地与各个 CE(包括它的主控 CE 和备份的 CE)建立联接之后,CE 将对 FE 进行初始配置,这个过程就是联接建立(图3中的 associating)的过程。

联接建立过程结束之后,CE 将激活 FE,使它可以开始转发分组,并且在 FE 与 CE 之间建立数据通道来传输控制层和转发层之间的用户数据。

联接建立过程结束后,CE 和 FE 间的信息交互负责维护系统状态,完成对 FE 的配置,用户数据业务转发,异常事件处理等功能。如图3所示,状态转换图中 CE/FE 状态包括 Active,Inactive,Down,Associated,引起状态转换的事件可以是 FE 接收到来自 CE,或 FE manager 的管理消息,CE 收到来自 CE manager 的管理消息,或者 CE-FE 间连接失败。在联接建立后的稳态通信过程中,FE 或者 CE 出现了不可恢复的错误,它就会发送"PE Down"消息并把自己转成"Down"状态。如果 CE 希望某个 FE 停止处理分组,它将发送"PE Inactive"消息给 FE,使 FE 变成"Inactive"状态。

3 分析

FACT 协议的主要特点是对网元的服务可扩展性,可延展性和高可靠性的支持。

3.1 服务可扩展性支持

3.1.1 *消息类型对服务可扩展性的支持* FACT 协议定义了许多用来在 CE 和 FE 间进行信息交互的消息类型,并且提供了多种服务参数,如图1所示,协议规定的消息类型最多可以达到512种之多,而且 FACT 协议中消息的定义采用变长的服务参数,可以支持新服务的增加,如我们可以定义一个新的消息类型提供对主动网络服务中代码下载的功能。

3.1.2 *对 FE model[4]的配置可以支持扩展新的服务* FACT 协议与 SNMP 协议比较相近,它提供一个基本的用来在 CE 和 FE 间进行消息交互的协议,并定义 FE 模型的概念,FE 和 CE 之间交互的服务参数实际上是依据 FE 模型而设定的。

FE model 的树形结构图如图4所示,这个模型是可以扩展的,FE model 中定义的 FE 的端口属性、转发功能选项、功能组件的顺序、功能组件的参数以及算法等都是可以配置的(包括增加,更新,删除等操作)。FE 的行为可以通过对模型的配置来定制。这种对 FE 模型的动态的配置功能可以通过 FACT 协议与 FE model 的交互操作来完成。比如,对图4所示的模型中的 capability 项中的 component 中增加一个代码下载的功能组件,再通过代码下载相关的消息交互来支持代码下载服务,从而实现新服务的扩展。

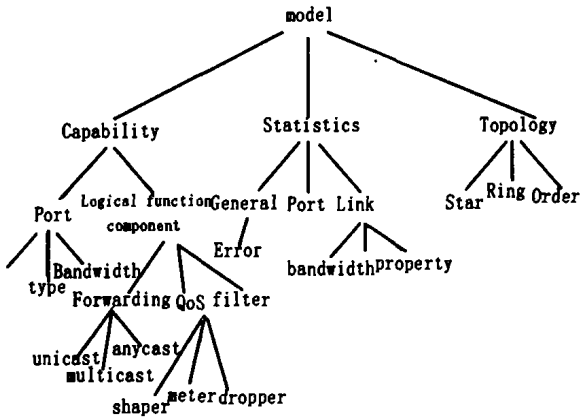

图4 FE model 的树图

3.2 可延展性(Scalability)支持

3.2.1 CE和FE的标号 如FACT协议消息头中所定义的,CE tag 和 FE identifier 最大都可以达到64k 个,这意味着一个网元最多可以包括64k 个 CE 和64k 个 FE。

3.2.2 拥塞控制对可延展性的支持 FACT 采用具有拥塞控制功能的传输协议,如 TCP,DCCP,SCTP。这些协议可以支持一个网元中成千上万个 CE 和 FE 同时通信所造成的拥塞。

3.2.3 动态联结和拓扑发现 当有成千上万个 CE 和 FE 时,对于一个管理员来说很难单独地对每个 FE 和 CE 进行管理。当一个 CE/FE 想要加入/离开一个网元时,FACT 协议中定义的动态联结和拓扑发现可以帮助它自动地加入和离开网元。

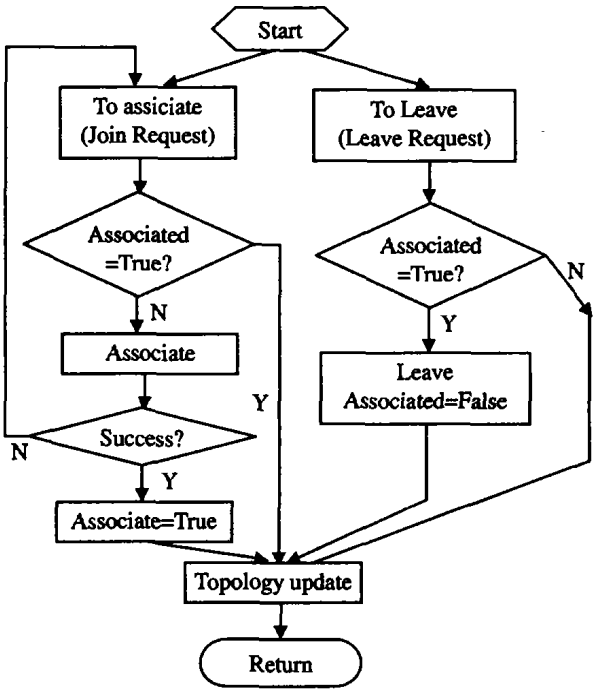

图5 动态联结和拓扑发现

如图5所示,在原型系统的状态转换图中,我们定义了一个"associated"状态来标识 CE 与 FE 的联接情况。如果 CE/FE 由于某种原因想要离开网元而且"associated"为真,它将发送"Leave request"消息;如果它想重新与 FE/CE 建立联接而且"associated"为假,它将发送"Join Request"消息。对方成功地收到消息并发送相应的响应之后,将更新这个 CE/FE 的状态信息,并更新网元内的拓扑信息。

3.3 对网元的高可用性支持

3.3.1 可靠性 当一个网元中的 CE 和 FE 相距超过一跳的距离,FACT 协议就必须要用可靠的传输协议,如 TCP/DCCP/SCTP 来为一些要求高可靠传输的消息提供保证,如关于 FE 配置的消息。

3.3.2 安全性 DoS 攻击的问题:攻击者可能会试图改变或伪造对网络元件的配置,或者通过传送无用的消息消耗掉 CE 和 FE 之间数据和控制通道的珍贵的网络带宽,比如无用的控制协议分组。这种攻击,比如一个关于 FE 间拓扑的错误的配置可能会导致网元工作错误或者延迟对重要的控制消息的处理。

FACT 协议需要有一个安全可靠的通信环境来避免 DoS 攻击。比如在 IP 连接中它要求用 TLS 协议在 CE 和 FE 间提供认证,而且还规定数据和控制消息的分开传输,保证控制消息的可靠传输,预防 DoS 攻击。

3.3.3 容错(fault-tolerant)的支持

1)事件报告。CE 对于 FE 向它报告的错误事件会进行处理,对 FE 的错误进行诊断和修复,对于不能修复的将启用备份(fail-over)。

2)Fail-over。FACT 协议为了增强网元的可用性采用了 CE fail-over 支持。当主 CE 出现不可恢复的问题时,备份的 FE 将接管对 FE 的控制权。过程如下:当 FE 发现一个主动 CE 出错后,它会向备份的 CE 发送异步的事件通知,然后由备份的 CE 对它执行控制和配置功能,并且 FE 把数据都转发到这个 CE。主 CE 与备份 CE 的配置比例可以是 N:1,1:1,或 1:N。

4 FACT 协议的应用

FACT 协议作为为 ForCES 框架定义的一种控制层和数据转发层之间的信息交互协议,可以用在采用 ForCES 框架的任何网络设备中,比如基于 NP 的路由器,如图6所示,网络处理器作为数据转发层的载体,即 FACT 协议中的 FE 的载体,控制层面可以采用通用处理器或者控制处理器作为控制层的载体,即 FACT 协议中的 CE 的载体,CE 和 FE 通过背板连接。从 FE(数据转发元素)发送到 CE(控制元素)的各种数据和控制管理信息都要经 FACT 协议的处理。采用 FACT 协议可以使得控制元素和数据元素即插即用,并且 FACT 协议支持冗余和扩展,使整个系统具备更强的可用性和可扩展性。

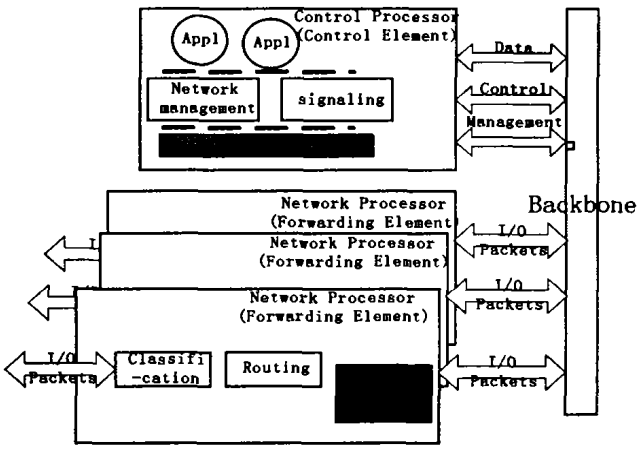

图6 FACT 协议应用于基于 NP 的路由器

5 FACT 协议的设计和原型实现

FACT 协议是一种工作在分布式环境中 master/slave 结构的协议。CE(master) FACT 协议为很多控制层的协议和软件提供服务,比如 OSPF,SNMP,RSVP 协议等。FE(Slave) FACT 协议接受 CE 的控制,并且为控制层用户转发分组,向 CE 报告异常事件和统计数据。

作为一个新方案,FACT 协议还没有任何实现。我们的原型系统是基于 Linux 内核2.4.18,在实现时采用的是基于 Linux 操作系统的 C 语言编程,为了使系统最后能与操作系统,连接网络都无关,对协议的实现采用了分层模块化设计。图7和图8分别给出了我们的原型实现的数据结构和功能模块图。

5.1 FACT 协议原型实现的数据结构

如图1消息格式所示,由于 FACT 协议的消息是变长的,它们不能够被直接映射成某种固定的数据结构,因此我们专

门设计了一种内部数据结构以方便协议消息的处理，如图7所示。从缓冲区中取出的协议数据要先经过解析成这种内部数据结构之后，才进行处理。这个数据结构也是协议引擎中与FE model及使用FACT协议的服务的其他实体之间进行交互的数据结构。

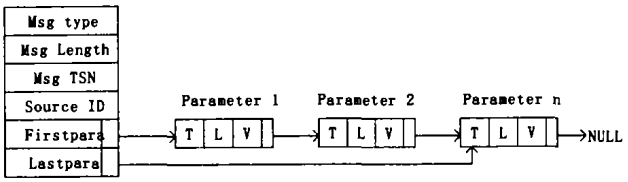

图7 协议引擎中使用的协议消息数据结构

5.2 FACT协议原型实现的主要功能模块

为了便于服务的扩展和协议的移植，我们采用了分层和模块化的设计方法，下层提供封装好的网络服务，上层是协议实体。主要包括3种模块：协议消息的发送和接收模块，协议消息处理模块（包括消息解析，各种协议消息类型的处理，封装），服务接口。

5.2.1 *消息发送和接收* 为了得到可靠的通信，作者采用TCP协议作为传输协议，并利用TCP提供的发送和接收服务，并用TCP连接建立控制和数据通道，分别用来传送FACT协议的控制消息和转发的用户数据。

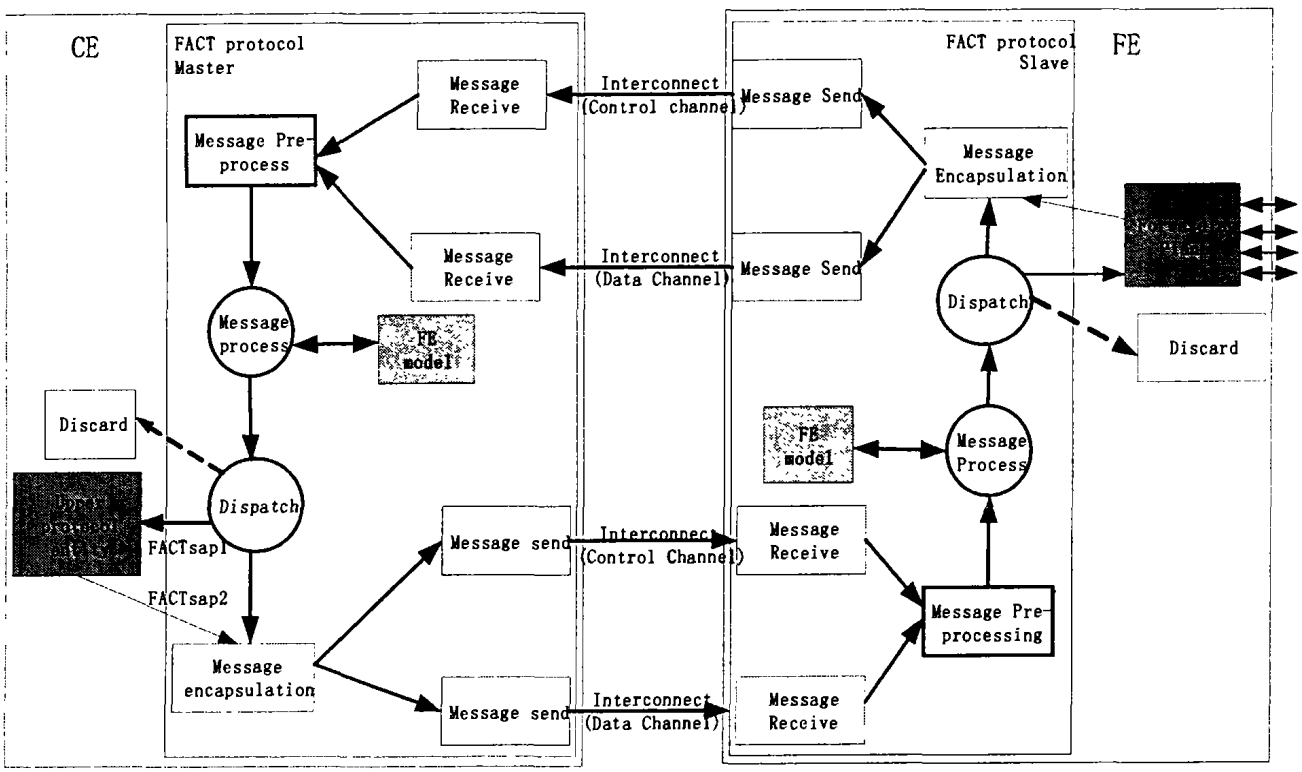

图8 FACT协议实现的功能模块设计

5.2.2 *协议消息的处理* FE/CE接收到的消息都要进行预处理（消息解析），把接收到的内容转换成内部的消息数据结构（如图7所示）以待进一步处理。经过预处理的消息被送给消息处理模块，消息处理模块将根据消息的类型把它送到相应类型的消息处理模块，比如，如果是向OSPF协议转发的分组将被送给OSPF的协议实体；如果是心跳检查（Heart Beat message），将简单地产生一个回应，封装上协议的消息头后将送到发送队列中发送出去；如果是需要更新FE的配置状态的消息，将对FE model中的LFB进行add，delete，update等操作；如果是收到了一个错误的消息，将丢弃消息。

5.2.3 *协议的服务接口*（FACTsap） 网元的控制层会用到FACT协议提供的服务的包括控制层的各种控制和管理协议和其它第3方软件，如图8中灰色方框中所示。这些实体从FACT协议中接收转发到它们的分组，并且把需要传送到别的网元的分组通过FACT协议转发到FE中，并通过FE转发出去。

UPEgetpacket(FACTsap1)，从FACT协议的服务接口中获取转发的分组。

UPEsendpacket(FACTsap2)，向FACT协议的服务接口发送需要FACT协议通过数据转发层转发到别的网元的分组或者发送到FE的命令。

6 相关的工作

FACT协议作为一个新的IETF ForCES工作组的草案，目前还没有其他的实现，但是它具有很大的潜力成为ForCES协议的标准。IETF ForCES[1]工作组为ForCES框架定义了一系列的信息交互协议，这些协议分别为ForCES框架[5]的不同的参考点提供信息交互，包括CE-FE，CE-CE，FE-FE，CE manager-CE，FE manager-FE，CE manager- FE-manager参考点等，其中CE-FE参考点的信息交互协议正在制定过程中，其他协议会稍后制定。目前CE-FE参考点的信息交互协议有3种，包括FACT[2]协议，Netlink2[6]和GRMP[7]协议。与其他两种协议相比较，FACT协议具有以下特点：

1）由于它采用了一个基本的信息交互协议，主要时间是对协议消息的处理，简单易实现。

2）重用了已有的协议来提供可靠性安全性支持，如TCP，SCTP，TLS等。

（下转第217页）

lo 进程如图3所示[3]。

第一步：确定非形式化测试目的。根据 SDL 模型（见图3），系统将收到三种消息，?begin、?IntervalTimer 和 !hello。输入输出对应着测试目的，所以系统的这部分模型中包含三个测试目的，即能够接收到 begin、IntervalTimer 和发送 Hello 消息。

第二步：划分测试组。先标记系统模型中的状态：系统中有两个状态，分别是 down 和 Wait_IntervalTimer。

再对测试组进行形式化描述，根据模型中状态的转换关系，将分为两个测试组：

1) down>>Wait_IntervalTimer

2) Wait_IntervalTimer>>Wait_IntervalTimer

以从 Wait_IntervalTimer 状态到 Wait_IntervalTimer 状态之间的测试例为例说明对测试组的描述：down;Wait_IntervalTimer>>Wait_IntervalTimer

第三步：对测试目的进行形式化描述。

down;Wait_IntervalTimer>>[IntervaTimer,Hello]>> Wait_IntervalTimer

这个步骤得到一个输入输出活动树，如图4所示。

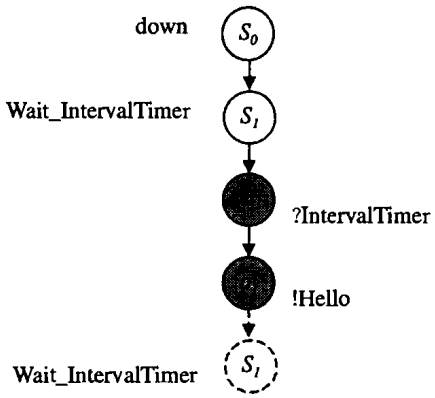

图4 发送 Hello 消息的输入输出活动树

第四步：生成测试例。

1. 获得描述中允许的进行的期望的活动路径：?IntervalTimer,!Hello

2. 这个例子中不需要补充，所以没有加入其它路径。

3. 根据输入输出活动树确定测试例中的测试步：?IntervalTimer,!Hello

4. 确定完成前导所需的输入输出序列

1)先确定前导状态序列：down，Wait_IntervalTimer

2)再确定相邻状态间状态的输入输出活动序列：?begin

从而推出前导输入输出活动序列为：?begin

将4中得到的前导输入输出序列分别和3中得到的两个测试例的测试步合并，就得到一个完整的测试例：?begin,?IntervalTimer,!Hello

第五步：转换角度。将测试例中的输入输出从 IUT 的角度转换到测试者的角度，即将接收?转换为发送!，发送!转换为接收?。

最终得到测试例：!begin,!IntervalTimer,?Hello

小结 本文提出的方法是通过一连串简单步骤来实现，简单的步骤在控制下保持自身的复杂性。保持测试套按照清晰的测试目的进行分组可以控制整体的复杂性。算法的设计并不局限于某个具体的协议。所以这个方法不仅能够完成 OSPF 协议的一致性测试套的生成，还能应用于其它的路由协议，比如 RIP 和 BGP 等路由协议。

本文提出了一种基于 SDL 和 MSC 模型的一致性测试的生成方法，它的优点如下：

1. 从协议的 SDL 模型和 MSC 模型出发产生的测试套能达到功能覆盖和数据覆盖；

2. 测试组和测试目的都是通过形式化方法进行描述，易于抽象；

3. 没有人为因素，利于保证测试例的正确性；

4. 测试例的选择容易执行；

5. 在整个生产生命周期中易于维护。

参 考 文 献

1 Information technology—Open Systems Interconnection -- Conformance testing methodology and framework -- Part 1: General concepts, ISO/IEC 9646-1,1994

2 Kerbrat A,Jeron T,Groz R. Automated test generation from SDL specifications. In: Proc. of SDL Forum '99, Montreal, Canada, 1999. 135～151

3 Deussen P H,Tobies S. Formal test Purposes and the Validity of Test Cases, FORTE2002,114～129

4 Ural H,Saleh K,Williams A. Test generation based on control and data dependencies within system specifications in SDL, Computer Communications, 2000,23:609～627

5 Robles T,Mañas J A,Huecas G. Specification and Derivation of OSI Conformance Test Suites, FORTE1993,177～187

6 Moy J. OSPF Version 2,STD 54, RFC 2328, April 1998

（上接第22页）

3) 高的可用性和可扩展性支持，如本文第3部分所介绍的。

4) 在消息定义中可以有8种优先级，提供了对 QoS 保证的支持。

5) 缺点是需要有一个安全可靠的运行环境来预防 DoS 的攻击。

总结 本文介绍了一种用在 ForCES[1]框架中的 master/slave 结构的控制协议 FACT[2]协议，它可以提供控制层对数据转发层的控制和配置，并且为网络设备提供服务扩展能力，可延展性以及高可用性的支持，对基于网络处理器构建可扩展服务的路由器，以及其他采用 ForCES 结构的网络设备都具有重要的意义。

参 考 文 献

1 http://www.ietf.org/html.charters/forces-charter.html

2 Audu A, et al. ForwArding and Control ElemenT protocol (FACT),draft-gopal-forces-fact-05. Sep. 2003

3 Khosravi H, et al. Requirements for Separation of IP Control and Forwarding. RFC 3654, Nov. 2003

4 Yang L, et al. ForCES Forwarding Element Model, draft-ietf-forces-model-01. Oct. 2003

5 Yang L, et al. Forwarding and Control Element Separation (ForCES) Framework, draft-ietf-forces-framework-13. work in progress, July 2003

6 Salim J H, et al. Netlink2 as ForCES protocol, draft-jhsrha-forces-netlink2-02. Nov. 2003

7 Wang Weiming, et al. General Router Management Protocol (GRMP) Version 1, draft-wang-forces-grmp-00. Nov. 2003