

DOI: 10.7652/xjtuxb201702002

互联网通用地址体系框架

朱亮, 徐恪

(清华大学计算机科学与技术系, 100084, 北京)

摘要: 针对地址体系导致互联网体系结构陷入僵化的问题,提出了一种通用地址体系框架模型,使得多种地址策略能够快速构建、部署、评估以及并存。该框架基于构件设计范型,通过细粒度的构造模块以及高度可配置的方式,允许应用按需检索、创建多样的地址范式,并支持在运行时共存。同时,通过抽象地址机制的交互模式,设计了一种能够涵盖地址体系核心特征属性的通用节点结构,并阐述了异构地址策略如何在该框架内灵活构建;实现了框架原型系统并对相关设计方法论进行了讨论。应用实验表明,通用地址体系框架在动态构建异构地址策略方面具备良好的通用性和可扩展性,相比传统互联网地址系统,能够更加高效地支撑互联网顶层生态的不断演化。

关键词: 互联网体系结构;地址体系;通用框架;构件设计;原型系统

中图分类号: TP393 **文献标志码:** A **文章编号:** 0253-987X(2017)02-0006-07

A General Framework of Internet Address Mechanisms

ZHU Liang, XU Ke

(Department of Computer Science and Technology, Tsinghua University, Beijing 100084, China)

Abstract: A general framework of Internet address mechanisms is proposed to solve the problem that the traditional address mechanism causes ossification of the Internet architecture and to allow multiple address strategies to be quickly built, deployed, evaluated and coexist. The framework bases on component based software engineering (CBSE) paradigm, allows the application to retrieve and create multiple address paradigms, and supports coexistence in runtime through the fine grained module and high configurable way. A scalable general node structure is designed through the interactive modes of abstract address mechanism, and the construction of the heterogeneous strategies is explained. A prototype of the framework is developed and its related design methodology is discussed. Test results show that the general address framework has good universality and scalability in dynamically constructing multiple address schemes. A comparison with the traditional Internet address system show that the framework supports the Internet ecological evolution more effectively.

Keywords: Internet architecture; address mechanism; general framework; component-based design; prototype

以 TCP/IP 为核心的互联网体系结构取得了巨大成功,既屏蔽了异构底层技术的多样性,也支撑着

收稿日期: 2016-10-07。 作者简介: 朱亮(1982—),男,博士生;徐恪(通信作者),男,教授,博士生导师。 基金项目: 国家自然科学基金资助项目(61170292,61472212);国家科技重大专项资助项目(2015ZX03003004);国家“863 计划”资助项目(2013AA013302,2015AA015601);国家“973 计划”资助项目(2012CB315803);欧盟 CROWN 基金资助项目(FP7-PEOPLE-2013-IRSES-610524);清华大学信息科学与技术国家实验室(筹)学科交叉基金资助项目。

网络出版时间: 2016-11-24 网络出版地址: <http://www.cnki.net/kcms/detail/61.1069.T.20161124.1855.004.html>

层出不穷的创新应用,保证了互联网的泛在连接和旺盛的生命力。然而,随着网络规模、应用需求的不断增长,传统互联网基于预定义固化协议分层的特性却限制了体系结构的可扩展性,静态功能锚点变化导致“牵一发而动全身”的效应,无法容纳体系结构内在机制的创新。比如增长地址空间就涉及到体系结构全局每个部件的改变,这种功能紧耦合的特性严重阻碍了互联网的演进。为解决互联网体系结构的僵化问题,业界提出了很多通用体系结构方案。可重构柔性网络^[1]通过将可重构技术引入网络构建,以实现网络服务的动态定制;可演化网络^[2]根据网络运行状态及应用特征进行自主调整,及时适应网络状况变化;4WARD 新型互联网设计方案^[3]是欧盟第七次框架计划下的未来互联网项目,目标是通过电信级的网络资源虚拟化实现多个网络并存,并灵活支撑商业环境下的可靠互操作,目前仍处于理论研究阶段;美国自然科学基金的 XIA 项目^[4]强调对多种通信场景的支持,对每一种通信规则,如主机、内容或服务分别定义一个细腰结构,用户通过制定适当的通信意愿,实现多种通信范式并存。上述研究旨在从宏观的体系结构层面实现互联网的可定制性,但是对命名、寻址、转发等机制都进行了特定限制,同时也存在设计空间过于庞大的问题,无法真正实现多维度的灵活性和通用性设计。本文主张在渐进演化原则的指导下,立足于体系结构某一特定核心部件,实现通用化以及可定制化改造,作为向体系结构层面衍生演化的基石,将有助于设计方法论的具体化和实用化。

事实上,地址是互联网体系结构中的核心组件,同时也是最具可塑性的部分,而体系结构其他组成部分如路由、传输控制等则依赖于地址组件的演化^[5-6]。因此,本文从地址系统角度出发,构建一种通用的体系框架以支撑多种地址策略的构建、部署乃至共存,异构的地址方案可以在统一体系框架中随时构建、灵活定制,这样可以为构造体系结构维度的设计奠定良好的基础,从而进一步指导体系结构的演进方向。

1 研究意义与设计原则

地址体系是互联网体系结构中的核心组成部分,其演进性决定了体系结构对上层网络多样应用需求的承载能力。未来互联网应能够根据应用需求,敏捷快速、动态创建多样的地址范式,并支持在运行时共存。

1.1 构建通用地址体系的必要性

IP 地址体系已成为阻碍互联网体系结构演进发展的主要约束常量^[7],导致了在扩展性、移动性以及安全性等方面面临诸多挑战。作为细腰的协议机制陷入僵化^[8],难以进行变革或创新,IPv6 的部署困境便是典型例证之一。而诸如 NAT、移动 IP 等修补机制极大增加了细腰处的复杂性,互联网得以持续演进的细腰模型正逐渐演变为人到中年的“水桶腰”窘境,学术界也开始了对互联网的重新思考。

尽管在互联网体系结构如何发展这一问题上存在着诸多争议^[9-10],但地址体系需要创新已成为学术界的一致共识。以信息为中心、面向语义分离的地址范式等诸多新型地址机制^[11-13]分别从特定维度对传统互联网模型进行了扩展甚至本质上的革新,但到底哪种地址机制设计更加优秀?其争论也从未停止过,比如基于主机或是内容命名的通信模型,扁平或层次化的命名机制,route-by-name 亦或是 lookup-by-name 的解析范式等等。事实上,上述地址相关机制的性能评估无法一概而论,在不同的通信需求场景下各有优缺点,机制之间需要灵活的组合并在统一框架内灵活构建、部署,才能获得进一步的定量评估。

另外,对于互联网应用需求的多样性来说,并没有一种“one size fits all”的地址机制,未来互联网也不应是一个细腰静态预定义的网络,而是可以按需裁剪定制以适应不同的通信需求。比如对于安全性较高的应用,可以动态构建特定安全机制并平滑集成到地址方案中;对于对功耗要求严格的应用,可以按需设计轻量级的命名寻址机制;针对基于内容的查找场景,可以灵活集成发布/订阅的地址范式。构建通用地址体系框架的意义在于允许多种地址范式并存以满足互联网顶层生态的不断演化,其示意图如图 1 所示。

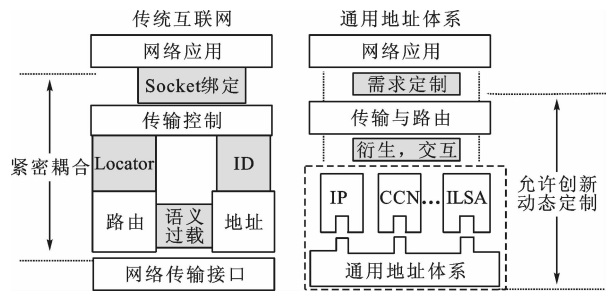


图 1 通用地址体系设计理念

1.2 设计原则与目标

从互联网的历史经验和发展现状来看,造成僵

化问题的根本原因在于固化的功能分层特性^[14],而不是分层原则本身。因此,各协议层内部的功能单元应具有更为灵活的组织形式(比如路由或者传输功能单元),以动态灵活地实现特定功能。因此,支持互联网地址体系自适应性的关键在于提供一套精心构造的可扩展模块化框架。基于构件的软件工程(component based software engineering, CBSE)^[15]由于其可重用以及可扩展等特性,与通用地址框架的设计理念完全契合,因此本文基于CBSE设计范型提出了一种通用地址体系支撑框架,异构的地址范式能够在该框架内快速构建、部署、评估,并支持运行时共存,为体系结构的演进多样性提供坚实基础,主要设计目标有以下几点。

(1)通用性:细粒度的构造模块以高度可配置方式形成特定功能组合,支持多样的命名、寻址解析、分组转发机制,赋予网络极大的灵活性以及自适应特性。

(2)灵活性:与传统互联网的分层结构以及粗粒度功能的严格集成特性不同,框架允许构件之间的跨层交互特性,并在特定场景下通过参数调整优化自身行为,有效构建地址策略的创新性实验。

(3)可扩展性:新的地址功能机制可以在该框架内动态增加、删除,以可插拔的形式根据需求而定制,允许安全、管理等机制的平滑集成。

(4)可伸缩性:网络节点由被动地传输信息发展为能主动地处理信息,用户能主动可伸缩地参与和定制自己需要的地址模型,并通过动态调整以构造所期望的行为。

本文以构件范型的设计方法论为线索,组织如下:①描述了通用互联网地址体系框架的总体模型;②对模型中的主要层面即构件容器层和地址实例层做了简要阐述,分别强调了地址构件如何抽象、设计,以及如何被组装形成特定地址机制;③讨论了通用框架的原型系统实现以及相关设计方法论。

2 通用地址体系框架

用地址体系设计的整体视图来看,应用和地址体系的功能单元能够协同工作,通过灵活选择、组合传输控制功能、命名寻址功能以及网络转发功能,并统一协调各功能单元行为以构建合适的地址范式,从而满足资源约束下应用的不同需求。图2描述了互联网通用地址体系框架模型,该框架包括以下组成部分。

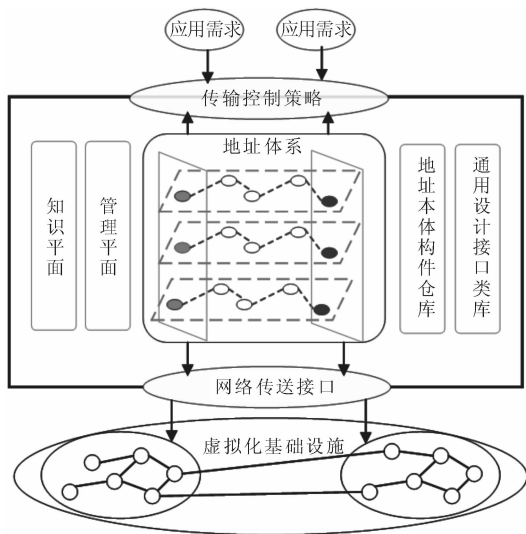


图2 通用地址体系框架模型

架,为便于阐述,可以从两种互为补充的视图进行阐述。水平层面与地址机制运行实例相关,从宏观的角度说明特定的地址机制实例如何通过虚拟节点或者分布在节点上的功能构件复合而成,称之为地址实例层。实例层由逻辑节点和链路连接而成,封装了地址策略的整体功能性,表明地址构件如何在领域模型及相关设计原则约束下在该框架内进行组装。地址机制设计遵循黑盒的模块化方式,每个节点或构件的具体实现封装在内部,而仅仅通过接口与外部交互;节点和构件由一组分布式的功能单元复合而成,逻辑地封装在竖直层面中。从微观的视角来看,竖直层面可看做是提供特定功能的构件容器,以支撑地址策略的设计,称为构件容器层。例如在设计地址命名机制时,可依据策略从命名容器中选择扁平或是层次化命名机制。竖直层面强调地址体系核心功能构件的识别、抽象以及具体的实现方法,各容器层中的构件能够灵活组合,定义通用的通信节点结构。连接实例层与容器层的关键元素是构件,容器层中的相同构件在不同实例层中可以有不同的实现。

(2)图2中左边代表一组与地址机制多实例管理和控制相关的功能,包括管理平面和知识平面。管理平面基于应用需求以及控制策略,进行构件选择、组装决策,并验证构件之间的依赖,优先约束以及接口兼容性,为构造地址机制实例提供一个优化的组装配置方案。另外,管理平面还可以通过控制接口对构件的接口参数进行操作,动态调整实例运行状态并进行性能监控;知识平面存储地址体系领域模型,提供设计规范约束,在装载地址实例之前验

证功能正确性,即是否满足先前的应用需求策略。另外,知识平面还维护网络的资源约束、拓扑状态、上下文信息一致性,以及方案对比、评估、以及最佳方案推荐等,为管理平面的组装配置提供更多的决策基础。

(3)为了实现最大的通用性和灵活性,本文所提互联网通用地址体系框架还集成了软件工程相关的设计方法论。图 2 中右边是由构件和通用设计接口构成的存储仓库,用来辅助地址机制的设计并进一步增强设计自由度。构件库是由构件、容器、构件管理程序 3 个部分构成,实现构件形式化描述、分类、检索等功能,提供一个统一的构件管理和使用平台。构件库随着框架的开发设计而不断完善,符合框架设计规范的构件、设计范例、安全、映射等机制等都可以动态加载到构件库中。通用设计类库包含地址体系中多维度的抽象接口和交互模型,提供用来构造地址系统过程处理模式的核心原语和类库。

(4)从应用角度来看,无需感知图 2 中底层基础设施的变化,应用只需创建一个符合规范的需求描述,交由管理平面进行解析,同时依托知识平面给出构件选择、组装方案,构造一种特定的地址机制并部署到虚拟化网络切片中。同时,利用构件式分层设计的跨层交互特性,地址机制允许与相应的传输控制构件进行功能交互或组合(比如流量控制,QOS 等),以形成完整的体系结构层面的服务体系支撑应用需求。

如果将体系结构看作是内核,那么本文所设计的地址体系框架则是一个最小化的微核。微核需同内核中的其他组件交互,共同完成整个体系结构层面的支撑功能,比如框架里并不指定与传输控制相关的策略,特定的地址方案可在具体设计时灵活定义与其他功能构件的交互行为。

3 地址体系通用节点结构

当前诸多的新型地址机制缺少统一而高效的部署试验环境,主要原因在于:①无统一设计规范和一致性交互语义,需要针对每种地址策略实现特定的构建环境,代价大且缺少灵活性,例如 NDNsim^[16]和 COREsim 网络模拟器^[17]分别针对 NDN 互联网体系结构^[18]以及 LISP^[19]的实现机制而构建部署实验环境,且互不兼容;②构造一个体系结构层次的原型试验环境代价很大,并且关注点太过宽泛,缺少针对性,比如在 NS3 仿真器^[20]中模拟非 IP 的命名寻址机制,则需要修改大量的源代码。为准确聚焦于

互联网地址系统的灵活构造和部署试验,提供了一种能涵盖地址体系核心特征属性的构件容器。该容器可看做是一种由构件组成的通用节点结构,构件包含多个功能单元用于实现具体地址机制相关的服务,比如命名、映射机制等。节点可以根据特定应用需求,灵活选择、组合容器内的构件,而新功能也能够以构件形式动态加入容器。事实上,通用节点结构就是本文提出的地址体系框架模型中容器层的实现,包括核心构件的识别、抽象以及相应实现。每个功能构件被封装在对应容器中,而容器则基于黑盒构造,因此具有设计和开发的独立性。对于一个虚拟节点而言,节点结构可以在通用节点结构的基础上被按需复制,继承或者扩展,在各个层面上提供可复用的数据结构和功能单元以加速地址原型的构建过程,形成即插即用的构件体系,快速灵活地构建多种地址服务策略。通过对多种地址机制交互行为的抽象,总结出以下通用节点结构:命名、解析、转发、存储以及状态模型,如图 3 所示。

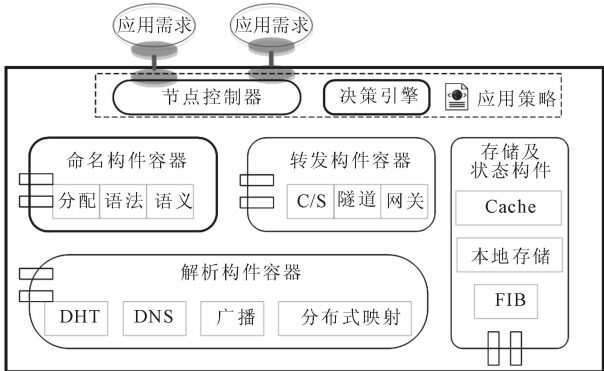


图 3 通用节点结构

3.1 命名构件

命名机制是地址体系中的核心部分,解析、寻址都需依赖命名机制进行。针对当前地址的缺陷,越来越多的地址命名类型被提出,无论是演进式还是革命式,其地址格式相对于 IPv4 都进行了相应创新,如基于加密产生地址的 CGA 网络体系结构^[21],基于内容命名的 NDN^[18]、基于公钥哈希的 HIP^[22]以及基于自我验证的地址 AIP 网络体系结构^[12]等,各地址类型的特征语义、构造方式等都有很大的不同。为保证构建地址策略的通用性,设计了一种支持多种命名方案的描述规范(generic address description language, GADL),通过总结各种互联网地址体系的结构特征,构造出一套语义完备、可解释当前乃至未来地址属性的描述规范,包括地址类型、地址长度、前缀、构造方式等核心属性,形成各种地

址类型的统一描述。

图 4 给出了一个 IPv6 地址语语义特征的简单描述。值得注意的是,上述地址配置文件只需在框架初始化时解析一次即可,目的在于使框架能够理解任意地址格式的语语义,并能够为指定网络节点或者自动批量命名,从而有效提高命名分配机制的灵活性。GADL 通过精心抽象的元素属性以及灵活的属性组合方式,提供了对地址模型的充分表达能力,既能无歧义描述现有模型,也能自定义新的模型创建新型地址范式。对于一些复杂的命名机制,如 ICN 以及一些基于密码学的地址产生机制,描述规范集成了一些灵活的 metadata 集合,如安全相关、内容 hash,加密算法、自认证、非对称密钥的机制等,以实现带有安全机制的复杂命名规范。

```
<!-- 定义命名空间 -->
<Gadl:namespace name= "Times Roma" >
  <Gadl:syntas><!-- 定义语语法结构 -->
    <!-- 地址长度 -->
    <Gadl:length>128</Gadl:length>
    <!-- 地址结构 -->
    <Gadl:structure type= "hierarchical" />
    <!-- 地址层级数 -->
    <Gadl:blocks>8</Gadl:blocks>
    <!-- 分割线 -->
    <Gadl:separator> </Gadl:separator>
  </Gadl:syntax>

  <!-- 地址从语义逻辑上的分级 -->
  <Gadl:semantic>
    <!-- 定义地址中用于路由的部分 -->
    <Gadl:LOC name space = "#">
      <Gadl:name>locator</Gadl:name>
      <Gadl:length>64</Gadl:length>
    </Gadl:LOC>
  </Gadl:semantic>
</Gadl:namespace>
```

图 4 基于 xml 的 IPv6 地址描述

3.2 解析构件

在当前网络中,名字和地址的概念以及他们应该如何映射解析并没有统一共识。比如一个协议层中的实体是如何引用另一个实体,两个实体之间的映射如何被识别等概念容易被混淆。基于本文提出的地址领域模型,解析发生在体系结构的多个层面上,是绑定过程的逆关系。绑定事先定义了不同语义范畴内对象的引用关系,形成了语义范畴的绑定拓扑,而解析实际上就是遍历语义范畴的过程。在这一设计原则指导下,解析构件容器除了能够提供常用的解析机制如 ARP、DNS 等,还允许自定义解析机制。自定义的解析机制只需实现按照规范,预先指定绑定拓扑,则解析过程则能够逆向自动实现。另外,基于语义范畴的解析原则使得在体系结构中引入新的

协议层变得更加容易,同样只需指定该协议层的语义范畴,并指明其在绑定拓扑中的关系即可。

3.3 转发构件

构建传输构件的的主要目的在于解耦合通信对象的内部行为与外部转发机制,实现转发构件的可重用,从而降低通信过程的复杂性。基于这样的考虑,抽取出连接两个对象的转发通道(channel)作为构件实例,通信对象的自身行为被封装,交互行为的语义规约到通道上,可作为独立于具体通信机制的多语义连接件使用。比如对于隧道实例而言,可通过赋予转发构件封装、解封装的行为语义时序而创建。在实际实现中,通道实例基于角色(Role)和粘合剂(glue)进行语义说明。Role 描述外部行为规范,而 glue 用于规定行为时序。每个转发构件都规定了通道不同的语义,来连接不同的通信主体。本文主要定义了一下 6 种转发构件实例:封装通道、隧道、网关通道、RPC 通道、直接转发、加密通道。对于其他转发场景,同样可以通过赋予 channel 行为语义时序,以灵活创建特定的转发构件。

另外,存储构件主要负责信息的存储机制,包括缓存,本地存储以及一些相应的算法;状态构件用于保存所需映射信息,比如 ARP 表、转发表等。对基于信息中心网络的地址策略^[20]来说,则需增加 PIT、FIB 表等状态存储。基于面向对象的方式,每个构件基于即插即用设计理念,保持接口和基本结构的内聚性,能够适应各种特定地址策略。控制器依据决策引擎选择这些构件以及管理组装的顺序,例如根据扁平或层次化的命名空间选择 DHT 或者 DNS 解析机制。如果需要增加一种新的构件,只需简单插入到相应容器,并在控制器里重新配置一下即可。

4 设计方法论及原型

通用地址体系框架基于构件设计范型,允许多种地址原型的快速构建,其构件式设计思想涵盖整个体系框架的生命周期,如图 5 所示。图 5 的上半部分说明了软件工程层面的构件化设计方法,下半部分描述了通用地址体系框架的原型系统结构。设计方法论用于指导原型系统的开发,而原型系统同样能支撑设计过程,并且通过消息反馈使设计得以迭代求精。

框架的生命周期从领域分析开始,经过抽象模型设计,再到构件的具体实现和组装,主要包括以下内容。

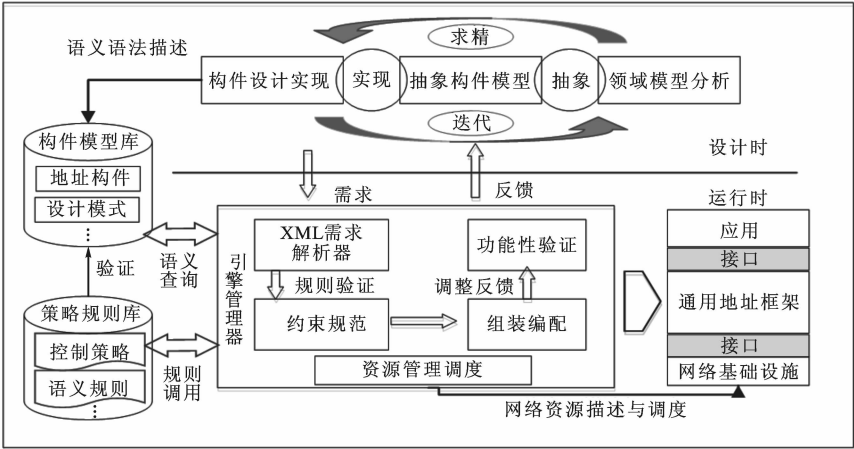


图 5 通用互联网地址体系原型及设计方法论

(1)领域分析:通过抽象特定领域的共性模式,对领域知识进行通用化抽象,并且确定构件粒度,例如转发功能是否应当作为一个构件而实现。该阶段实际上是容器层的抽象构件视图。如果初始需求分析出现了一种新特性,那么后续的构件设计就必须考虑到该功能的集成。

(2)抽象构件模型:在领域分析基础上,从宏观层面衍生出特定功能的识别和定义,以及如何被复合或者如何分布在节点上。比如命名、解析、转发等构件的组装顺序等。该阶段的输出是容器层和实例层的规范性描述。

(3)构件设计实现:该阶段主要关注实现技术细节包括接口性质、规范、描述等,输出是容器层构件的具体实现,实现的构件将自身的语法语义描述在构件库中注册。

上述设计方法论同样也是贯穿在本文中的叙述线索,整个设计过程是迭代的,通过精化中间模型,以确保最终地址机制满足所期望状态。图 5 的下半部分表示框架原型的实现部分。本文基于 Visual Studio 初步实现了原型设计,主要包括构件库与策略库的设计、解析引擎管理器以及用于预组装构件的编辑器。xml 解析器负责对应应用需求进行解析,并在相关控制策略的约束下使用基于语义的方式检索构件库。构件库响应调用,返回对应的构件描述。地址构件的组装过程在初始阶段或运行时需求约束变化而触发。构件选择器接收应用创建的服务请求,该请求以一种语法语义规范的格式定义,比如基于 WSDL^[23]的需求描述。之后选择器对服务请求进行解析,到构件库中查找是否存在已经满足此需求的解决方案,如果没有再检索满足需求的构件。检索完成后将所需装配的构件集合提交至构件组装

引擎。组装引擎依据控制策略以及一组自定义的算法^[24],将构件集合转化为构件组装方案,动态链接到需要的构件模块,并且为其创建实例化状态。组装引擎在实际部署前验证构件组合的一致性、优先约束等,例如一个构件的前置需求是否得到满足。该验证过程通过逻辑组合的蓝图实现,即事先进行逻辑意义上的构件组装,允许在实际部署前对地址体系进行模拟和验证,以寻找一些潜在的错误和冲突。另外,这种预组装特性能自动聚合构件性能,比如解析时间、转发延迟等,这样机制设计者能够事先预估地址体系的性能,并在部署之前评估不同的地址方案。

原型系统提供了 3 种层次的构件设计视图:功能单元设计、实体组装以及拓扑配置。系统通过提供 object、message 等核心类库及相关交互模式,以允许用户继承或扩展这些核心基类实现所期望的功能单元(如缓存、解析等构件),并以列表形式供用户按需调用。

基于该原型系统,在 1 000 个网络节点的规模下,分别针对 IP、LISP、NDN 这 3 种典型地址机制,从需求处理(检索)时间、命名规范解析与分配时间以及构件组装时间 3 个方面进行了测试(如表 1 所示),实验表明,通用地址体系平台的性能和可扩展性均能达到预期设计目标。

表 1 地址策略构建时间测试分析

地址策略	需求处理 时间/s	命名解析 时间/s	构件组装 时间/s
IP	0.02	0.05	0.18
LISP	0.06	0.07	0.14
NDN	0.11	0.04	0.23

5 结 语

固化的功能紧耦合特性难以适应动态多变的应用需求已成为当前互联网所面临的主要矛盾之一,而地址体系作为体系结构的核心组件是导致体系结构陷入僵化的主要约束锚点。为了解耦互联网严格的功能固化问题,本文基于构件设计范型,提出了可容纳多种异构地址机制的统一框架,并以设计方法论为叙述主线,分别讨论了地址体系框架模型及节点结构,最后给出了该框架的原型系统以及基于该原型的设计过程。通用地址体系框架立足于地址这一核心部件,实现了通用化以及可定制化改造,不仅能够统一框架内快速创建多样的地址范式,并支持部署和评估,更重要的是,通用地址框架可作为向体系结构层面衍生演化的基石,将有助于互联网体系结构设计方法论的具体化和实用化。在下一步工作中,我们将深入研究针对地址体系特点的虚拟网络资源调度算法以及高效的构件组装机制,并对原型系统进行迭代开发,使通用地址体系框架能够得到进一步完善。

参考文献:

- [1] 刘强,汪斌强,徐恪. 基于构件的层次化可重构网络构建及重构方法 [J]. 计算机学报, 2010, 33(9): 1557-1568.
- LIU Qiang, WANG Binqiang, XU Ke. Construction and reconfiguration scheme of the hierarchical reconfiguration network based on the components [J]. Chinese Journal of Computers, 2010, 33(9): 1557-1568.
- [2] 刘涛,钱德沛,王锐,等. 一种可演化网络的研究与实践 [J]. 西安交通大学学报, 2008, 42(10): 1193-1203.
- LIU Tao, QIAN Depei, WANG Rui, et al. Research and practice on evolutionary networks [J]. Journal of Xi'an Jiaotong University, 2008, 42(10): 1193-1203.
- [3] NIEBERT N, BAUCKE S, EL-KHAYAT I, et al. The way 4WARD to the creation of a future internet [C]//IEEE International Symposium on Personal, Indoor and Mobile Radio Communications. Piscataway, NJ, USA: IEEE, 2008: 1-5.
- [4] HAN D, ANAND A, DOGAR F, et al. XIA: efficient support for evolvable internetworking [C]//Proceedings of the USENIX Conference on Networked Systems Design and Implementation. Heidelberg, Germany: Springer-Verlay, 2012: 817-826.
- [5] BALAKRISHNAN H, LAKSHMINARAYANAN K, RATNASAMY S, et al. A layered naming architecture for the Internet [J]. ACM SIGCOMM Computer Communication Review, 2004, 34(4): 343-352.
- [6] SPYROPOULOS T, FDIDA S. Future Internet: fundamentals and measurement [J]. ACM SIGCOMM Computer Communication Review, 2007, 37(2): 101-106.
- [7] AHLGREN B, BRUNNER M, EGGERT L, et al. Invariants: a new design methodology for network architectures [C]//Proceedings of the ACM SIGCOMM 2004 Workshops. New York, USA: ACM, 2004: 65-70.
- [8] ALVES S. The Internet balkanization discourse backfires [J]. Social Science Electronic Publishing, 2014, 55(9): 1-8.
- [9] REXFORD J, DOVROLIS C. Future Internet architecture: clean-slate versus evolutionary research [J]. Communications of the ACM, 2010, 53(9): 36-40.
- [10] FELDMANN A. Internet clean-slate design: what and why? [J]. ACM SIGCOMM Computer Communication Review, 2007, 37(3): 59-64.
- [11] RAMIREZ W, MASIP-BRUIX X, YANNUZZI M, et al. A survey and taxonomy of ID/locator split architectures [J]. Computer Networks, 2013, 60(2): 13-33.
- [12] ANDERSEN D G, BALAKRISHNAN H, FEAMSTER N, et al. Accountable Internet protocol (AIP) [C]//Proceedings of the SIGCOMM 2008 Conference on Data Communication. New York, USA: ACM Press, 2008: 339-350.
- [13] CLARK D, BRADEN R, FALK A, et al. FARA: reorganizing the addressing architecture [C]//Proceedings of the ACM SIGCOMM Workshops. New York, USA: ACM, 2003: 313-321.
- [14] ATKINSON R, BHATTI S, HAILES S. ILNP: mobility, multi-homing, localised addressing and security through naming [J]. Telecommunication Systems, 2009, 42(3/4): 273-291.
- [15] KOZACZYNSKI W, BOOCH G. Component-Based software engineering [M]//Component-based Software Engineering. Berlin, Germany: Springer, 2004: 34-36.
- [16] MASTORAKIS S, AFANASYEV A, MOISEENKO I. NDNSIM 2.0: a new version of the NDN simulator for NS-3 [EB/OL]. [2016-07-25]. <http://119.90.25.25/named-data.net/techreport/ndn-0028-1-ndnsim-v2.pdf>.

(下转第 78 页)