计算机研究与发展
Journal of Computer Research and Development
DOI:10.7544/issn1000-1239.2017.20151139
54(5):940-951,2017

一种形式化的互联网地址机制通用框架

朱 亮 徐 恪 徐 磊
(清华大学计算机科学与技术系 北京 100084)
(tshbruce@gmail.com)

A Formal General Framework of Internet Address Mechanisms

Zhu Liang, Xu Ke, and Xu Lei
(*Department of Computer Science and Technology, Tsinghua University, Beijing* 100084)

Abstract The address mechanism is the most essential and important part of the Internet architecture and its evolution determines the capacity of the Internet to accommodate the innovative applications. The traditional IP-based address strategy gets the current Internet into ossification which makes the architectural innovation become a consensus. Many novel address strategies make significant extensions or innovations for the traditional model but lack of common design principles and consistent expression model. It has become difficult to insight into future evolution progress of the address schemes for the diversity and heterogeneity. Moreover, we believe that a diversity address mechanism might coexist in the Internet architecture to meet the ecological evolution of network applications. To tackle the above problems, by researching the evolution of the Internet address mechanisms and abstracting a minimal architectural core, a general framework for accommodating the diversity and heterogeneity of various address strategies is proposed in this paper, including: 1) formal and verifiable conceptual model forms a consistent theoretical framework within which the invariants and design constraints can be expressed; 2) abstract multi-dimensional and extensible interface primitives and interactive patterns with the communication axioms to provide a proof framework for the Internet address schemes; 3) derive working prototype implementations—Universal Engine of Address Schemes which allows us to construct the various address mechanisms with flexibility and support the evaluation, evolution and coexistence of the Internet address strategies, in order to meet the ecological evolution of network applications.

Key words Internet architecture; general framework; formalism; address mechanisms; correctness proof

摘 要 地址机制作为互联网体系结构中的核心组成部分,其演进性决定了对上层网络创新应用的承载能力.传统IP地址的缺陷导致当前互联网陷入僵化,大量新型地址机制的异构性使研究者很难以统一

收稿日期:2015-12-20;**修回日期**:2016-04-19

基金项目:国家自然科学基金面上项目(61170292,61472212);国家科技重大专项基金项目(2015ZX03003004);国家"八六三"高技术研究发展计划基金项目(2013AA013302,2015AA015601);国家"九七三"重点基础研究发展计划基金项目(2012CB315803);欧盟CROWN基金项目(FP7-PEOPLE-2013-IRSES-610524);清华信息科学与技术国家实验室(筹)学科交叉基金项目
This work was supported by the General Program of the National Natural Science Foundation of China (61170292, 61472212), the National Science and Technology Major Project of China (2015ZX03003004), the National High Technology Research and Development Program of China (863 Program) (2013AA013302, 2015AA015601), the National Basic Research Program of China (973 Program) (2012CB315803), the EU Marie Curie Actions CROWN (FP7-PEOPLE-2013-IRSES-610524), and the Multidisciplinary Fund of Tsinghua National Laboratory for Information Science and Technology.

通信作者:徐恪(xuke@mail.tsinghua.edu.cn)

方法论解释和把握未来互联网地址体系的演进发展.针对上述问题,通过对互联网地址机制的演化进行深入研究,抽象最小化核心特征,提出一种能够容纳异构地址策略构建乃至并存的通用框架,包括:1)完备的形式化概念模型,赋予地址常量的精确定义,并形成相关设计原则及约束规范的一致性理论基础;2)抽象多维度、可扩展的接口原语以构建3种核心交互模式,并结合通信公理化性质以及语义,构造一个地址交互过程的正确性证明框架;3)推导出通用地址引擎原型,允许灵活构建地址策略,支持异构地址机制的评估、演进以及共存,以更好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;通用框架;形式化;地址机制;正确性证明

中图法分类号　TP393

以IP地址机制为核心的互联网体系结构取得了巨大成功,既能兼容异构的底层技术,也支撑着层出不穷的网络应用,保证了互联网的泛在连接性和旺盛的生命力.随着互联网与经济、社会以及传统行业各领域的深度融合,“互联网+”代表着一种新的社会经济形态而诞生,以互联网为纽带催生了大量的创新应用,全面提升了社会发展力[1].

然而,在网络应用繁荣发展,规模不断增长的同时,作为基础设施的互联网体系结构正面临着扩展性、移动性以及安全性等方面的一系列挑战,限制了对上层新型应用的承载能力,学术界开始基于应用的适应能力来评估互联网的可演进性[2-4].同时,研究者们也开始了对互联网的的重新思考,并认为传统地址机制的缺陷,诸如语义过载等问题是限制体系结构演进的主要原因,而IP地址是造成互联网僵化的主要隐式约束[5].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[6-7],但地址机制需要创新已成为学术界的一致共识,大量修补或革新机制被提出,例如NAT(network address translation)、IPv6地址过渡策略、大量的地址安全[8]以及空间分离等方案[9-13].上述地址相关机制对传统互联网进行了扩展甚至本质上的革新,然而却缺乏统一的指导原则和理论基础,导致了设计的随意性,从而难以进行一致性的分析和评估.同时,缺少一种定义完备的通用模型去表述和解释这些机制,一些基本概念如名字、地址等都无法清晰定义,更不用说寻址、映射等复杂交互过程,其异构性和多样性导致我们很难以一种统一的方法论把握未来互联网地址体系的演进和发展.另外,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部分,地址体系的灵活性决定了体系结构中其他组件的可演进性[14].因此,我们认为,固定的地址机制或许难以支撑未来互联网需求,未来可能是多种策略并存以满足互联网顶层生态的不断演化.

针对上述问题,本文介绍了一种能够容纳地址机制多样性的通用地址框架,异构的地址机制都可以利用框架内定义的概念与过程交互模型构建部署.该框架通过一个精炼的最小化静态核心而独立于具体实现细节,从而满足最大限度的设计灵活性.同时,通用模型提供了相关设计原则的约束规范以及过程交互的正确性证明,保证了地址机制定义的一致性和准确性.本文对当前地址体系演进进行了深入研究,提出了:1)完备的形式化概念模型,包括相关设计原则约束,赋予地址体系的一致性定义;2)抽象多维度的转发、交换接口原语,基于这些可扩展的标准接口,可以灵活定义3类基本交互模式以构造具备正确性验证机制的地址抽象通信过程;3)基于概念和交互模型,推导出通用地址引擎作为该通用框架的原型系统,支持地址机制的演进以及多样性共存,以更好地支撑顶层的网络应用.

1　抽象通信模型

在介绍概念框架之前,我们首先引入一种抽象通信模型.该模型抽象了地址机制的核心组件及其通信过程,可独立于具体实现策略去研究分析,并推导出一些重要通信公理化性质,作为概念模型以及通信正确性证明的理论基础.

1.1　模型定义

抽象通信模型如图1所示.

我们首先定义5个概念:

1)消息(message).通信中的抽象数据交换单元.在实际网络中,消息可实例化为分组或MAC帧等具体格式.对于该通信模型,我们只关注其抽象结构而非比特层面的视图.一般来说,消息包含了一组头部域(源信息和目的信息等)用来指示转发状态,我们将这样的头部域称之为转发指令(forwarding

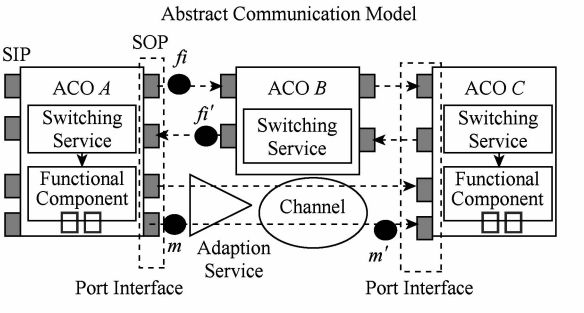


Fig. 1 Abstract communication model

图 1 抽象通信模型

instructions, fi), fi 可在消息传递过程中被特定网络实体重写.

2) 抽象通信对象(abstract communication object, ACO). 参与网络通信的一般性抽象对象,也是原型系统实现的核心抽象类,以下简称对象. ACO 可理解为网络中的一层协议栈或是交换引擎等通信组件. 除了消息传递,ACO 内部还包含一些功能组件对输入的消息进行处理. 实际的网络实体可通过继承 ACO 并与其他子组件(缓存等)通过特定机制聚合而产生.

3) ACO 通过端口(port)进行通信. 消息 m 在 ACO A 的输入端口(sink ports, SIP)和输出端口(source ports, SOP)接收和发送,可分别表示为 $m?A^{\text{SIP}}$ 以及 $m!A^{\text{SOP}}$. 端口不仅表示物理的网络接口,还定义了 ACO 在外部逻辑交互点上的操作. 所有端口的操作行为构成了 ACO 的接口域(interface),规定了该 ACO 的外部行为规范.

4) ACO 内部通常维护着一组映射表(local switching table),交换服务(switching service)通过查询本地映射表或是第三方 ACO(如分布式映射系统),根据返回的结果对转发指令 fi 进行相应处理. 交换过程通过执行〈抽象对象,转发指令〉二元组之间的映射以指明消息的后续转发状态,对象 B 查找本地交换表获取 A 和 C 的映射关系可表示为 $SP_B(A,fi):\langle A,fi\rangle \mapsto \langle C,fi'\rangle$.

5) 端口通过逻辑链路相连,ACO 之间所有的逻辑链路组成通道(channel). 通道规定了 ACO 的角色(role)以及交互规则和行为规范. 实际通信中,考虑 $O_A \backslash I_C \neq \varnothing$ 即 A,C 的行为语义不匹配,若想实现通信,则需要将不匹配的行为映射到 C 的输入域中,称之为适配服务(adaption service),可表示为 $AP(m):m \mapsto m'$(消息结构层面上的适配). 适配服务可存在于中间件,考虑到通道的可重用性,我们定义适配服务逻辑上由通道提供.

为更清晰阐述该模型的组合语义,下面采用 BNF 范式对抽象通信模型的组件对象进行形式化规约,其中:"〈…〉"表示非终结符,"[…]"表示出现 0 次或者多次,"[…]+"表示出现 1 次或者多次.

ACO::=〈Interface,Switching〉;

Interface::=[〈SIP_BEHAV〉]+[〈SOP_BEHAV〉];

SIP_BEHAV::=[〈Message〉]+[〈Sink_Operation〉];

SOP_BEHAV::=[〈Message〉]+[〈Source_Operation〉];

Switching::=[〈Message〉]+[〈Switching Service〉];

Channel::=[〈Role〉]+[〈Adaption〉].

对象 ACO 的行为由接口域和交换行为构成. 接口域是输入和输出端口对消息操作原语的并集,定义了消息在端口上的转发操作,如发送、接收、调用等;交换服务描述了转发指令层面上的交换行为,通过查询映射表获取相应的 fi,以指明该消息后续的转发状态;当上述过程完成之后,适配服务根据获取的转发指令对转发的消息进行适配处理,并依据通信角色输出到相应的实例化通道上.

基于该抽象模型的语义描述,对于特定对象而言,1 次抽象通信可被表述为基本通信组件基于 3 种消息操作的交互过程,即交换(switching)、转发(forwarding)、适配(adaption). 抽象组件可通过聚合、继承而构造具体的通信实体,实现了模型的可扩展性;而交互过程的抽象保证了不同类型操作模式的松耦合以及模块化特性,以便本文更加清晰地解构互联网地址机制中复杂的交互过程.

1.2 基本通信性质

通过对该基本通信模型的观察,我们推导出以下重要通信公理化性质,其中→表示事件发生的先后顺序.

性质 1. 直接转发(direct forwarding).

$$\forall \text{ ACO } A,C,m \Rightarrow m!A^{\text{SOP}} \rightarrow m?C^{\text{SIP}}.$$

性质 2. 交换(switching).

$$\forall \text{ ACO } A,C,fi:\exists \langle C,fi'\rangle \in SP_B(\langle A,fi\rangle) \Rightarrow fi?A^{\text{SIP}} \rightarrow SP_B(\langle A,fi\rangle) \rightarrow fi'!A^{\text{SOP}}.$$

性质 3. 传递性(transitivity).

$$\forall \text{ ACO } A,B,C,m,m':\exists (m!A^{\text{SOP}} \rightarrow m?B^{\text{SIP}}) \wedge (m'!B^{\text{SOP}} \rightarrow m'?C^{\text{SIP}}) \Rightarrow m!A^{\text{SOP}} \rightarrow m'?C^{\text{SIP}}.$$

性质 4. 可达性(reachability).

$$\forall \text{ACO } A,C,fi,m,m':\exists SP_A(\langle A,fi\rangle),AP(m),(m!A^{\text{SOP}}\rightarrow m?B^{\text{SIP}})\wedge(m'!B^{\text{SOP}}\rightarrow m'?C^{\text{SIP}})\Rightarrow m!A^{\text{SOP}}\rightarrow SP_A(\langle A,fi\rangle)\rightarrow AP(m)\rightarrow m'?C^{\text{SIP}}.$$

性质 5. 可返回性(returnabilty).

$$\forall \text{ACO } A,C,m:\exists m!A^{\text{SOP}}\rightarrow m?C^{\text{SIP}}\Rightarrow m!C^{\text{SOP}}\rightarrow m'?A^{\text{SIP}}.$$

性质 1 表明了对象间的直接通信,消息从 A 的输出端口发送,在 C 的输入端口接收.性质 2 描述了本地交换行为,B 接受含有转发指令 fi 的消息,查找 fi 的映射关系,根据返回结果对 fi 进行相应操作,再将含有 fi' 的消息输出到端口进行转发.SP_B 返回一组结果,即一对多映射的情况,可涵盖网络中多宿主(网络接口对应多个 IP 地址)以及移动性设计(身份标识对应多个位置标识)等实际场景.性质 3 说明了转发过程的可复合特性.性质 4 通过性质 2 与性质 3 的结合,表明了消息的可达性,即经过交换、适配、转发复合后可到达的一组对象集合.性质 5 说明直接通信的逆向可返回性.

抽象通信模型针对转发指令 fi 的不同操作模式,抽象出基本交互模式,涵盖了地址体系的核心通信过程.事实上,上述基本性质同样可归纳为 2.1 节中提到的核心抽象过程,即交换、适配和转发.该模型可实例化为任何一种地址机制的交互场景,为了使结构更加清晰,本文将基于上述抽象模型和通信性质,从与转发指令实例相关的概念模型以及组件间交互的动态过程框架 2 个方面分别进行阐述和讨论.

2　地址机制的统一概念模型

互联网地址机制自设计以来,一直存在着相关概念定义的争论[15],基本通信术语缺少统一、明确的定义.另外对于一些过程模式如绑定、寻址等存在着概念层面上的歧义和不一致性.为保证通用的设计原则和理念,我们将针对以上概念提出一套合理、统一的解释模型,并基于集合论以及关系代数赋予了形式化说明.基于该模型进行地址标识机制设计时,设计理念可被精确且一致地表达,同时相关设计原则可依据模型进行规范性检查.

2.1　命名空间

为了对分布式系统中的实体进行区分和访问,有必要给其中的对象分配标识.命名(naming)就是给互联网的用户、设备、服务和数据等 ACO 分配与其自身相关联的语法符号(symbols).命名空间(namespace)是由特定语义范畴(semantic scope)内的命名关系构成的集族,指标集为$\{aco,symbols\}$.语义范畴内的通信对象遵循相同命名规范,满足以下定义.

定义 1. 语义范畴.

$$\{aco\mid(fi!A^{\text{SOP}}\rightarrow fi'?C^{\text{SIP}})\wedge(\langle C,fi'\rangle\neq\varnothing\in SP_B(\langle A,fi\rangle))\wedge(fi=fi')\}.$$

其中,$(fi!A^{\text{SOP}}\rightarrow fi'?C^{\text{SIP}})$ 表示转发指令 fi 被改写为 fi' 后,从 A 发送到 C 的输入端口上.$\langle C,fi'\rangle\neq\varnothing\in SP_B(\langle A,fi\rangle)$ 表明了 B 查找通信对象 A,C 的映射关系,且结果不为空.结合通信性质 2,4 可知,具备可达性,并且操作相同协议头部的一组通信对象集合构成了特定的语义范畴,如图 2 所示:

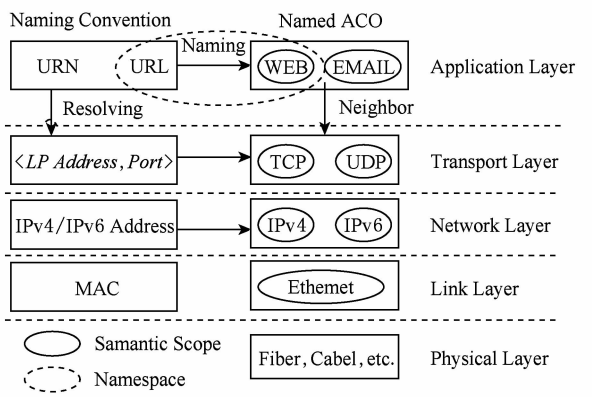


Fig. 2　Naming and resolving in the current Internet

图 2　当前互联网命名解析层次式模型

例如当前互联网协议栈中 HTTP ACO 与 IP ACO 则分别属于 2 个不同的语义范畴,从而关联着不同的命名空间.命名空间应被看作$\langle aco,symbols\rangle$的集合,命名对象决定了命名空间语义上的概念范畴,即标识何种类型或何种属性;命名规范则体现了语法上的表现方式.我们引入集合:可命名对象 $\mathcal{O}$,属于特定 semantic scope 的对象 $\mathcal{SO}$,任意符号集合 $\mathcal{S}$,其中 $\mathcal{SO}\in\mathcal{O}$.则命名空间定义如下:

定义 2. 命名与命名空间.

$$\exists aco:\mathcal{O},n:\mathcal{S}(aco\mathcal{N}n\Leftrightarrow(aco\mapsto n)\in\mathcal{O}\times\mathcal{S})\{aco\mathcal{N}n\mid aco\in\mathcal{SO}\wedge n\in\mathcal{S}\}.$$

命名可以使用二元中缀关系 $aco\mathcal{N}n$ 来表示,即将 n 作为对象 aco 的名字,是一种多对多的关系,即 1 个对象可以有多个名字,1 个名字可以标识一组对象,分别对应于当前互联网中的多宿主以及组播.值得注意的是,这里所说的命名是广义上的概念,其上下文依赖其语义范畴:即不仅是针对名字分配,还包括对身份、位置等抽象属性的命名,例如对象名字与

其附着点(attachment point)的命名空间分别被称为名字空间和地址空间(address space).

2.2 绑定、解析和寻址

命名空间的语义被限制在一定概念范围内导致不同语义范畴内的通信组件无法直接交互.为实现通信,绑定(binding)提供了访问异构命名空间的入口,也就是说,绑定关系 $\mathcal{B}$ 实际上构造了基于 semantic scope 的拓扑,为避免解析死循环,该拓扑应为有向无环图(DAG).绑定关系存在于不同语义范畴中的抽象通信对象之间:

定义 3. 绑定.

$$\forall aco_x:\mathcal{SO}_x, aco_y:\mathcal{SO}_y \cdot (aco_x \mathcal{B} aco_y \Leftrightarrow (aco_x \mapsto aco_y) \in \mathcal{SO}_x \times \mathcal{SO}_y) \wedge (\mathcal{SO}_x \neq \mathcal{SO}_y).$$

从命名空间的角度而言,Saltzer 认为绑定引用了名字和对象之间的关系[16],而根据上述定义,绑定产生于不同命名空间之间,可描述为 $\exists aco_x \mathcal{N} n_x$, $aco_y \mathcal{N} n_y \cdot \langle aco_x, n_x\rangle \mapsto \langle aco_y, n_y\rangle$. 当前互联网 TCP/IP 体系中,〈HTTP ↦ IPv4〉以及〈IPv4 ↦ Ethernet〉的绑定关系分别存储在 DNS 系统和 ARP 表中.如图 2 所示,如果 2 个 semantic scope 组件之间存在绑定关系,则满足通信性质 1,我们称之为邻居(neighbors).

由通信性质 5 可知,解析(resolving)可看作是与绑定具有相同基数(cardinality)的逆关系,包含所有有序对$\langle aco_x, n_x\rangle \mapsto \langle aco_y, n_y\rangle$的转置矩阵,表示为 $\mathcal{B}^{-1} = \langle aco_y, n_y\rangle \mapsto \langle aco_x, n_x\rangle$. 对于$\langle aco_y, n_y\rangle \in \mathcal{SO}_y$ 的解析过程定义如下:

定义 4. 解析.

$$\forall (aco_x \mapsto aco_y) \in \mathcal{B}; \beta(\langle aco_y, n_y\rangle \in \mathcal{SO}_y) = \{\langle aco_x, n_x\rangle \in \mathcal{SO}_x, e\}.$$

解析过程返回一组有序对的集合,e 表示指向下一个 semantic scope 入口引用.事实上,解析也是邻居发现的过程.对绑定拓扑进行递归或者分布式的解析过程为寻址(addressing),目的是为了获取最终可用于转发的标识符.结合通信的可传递性(性质 4),引入 2 种关系概念.

1) 关系复合.对于任意二元关系 $R: X \leftrightarrow Y$, $S: Y \leftrightarrow Z$ 来说,$R \circ S = \{x:X; z:Z \mid (\exists y:Y \cdot (xRy \wedge ySz)) \cdot x \mapsto z\}$;例如$(x \mapsto y) \in R$, $(y \mapsto z) \in S \Rightarrow (x \mapsto z) \in R \circ S$.

2) 自反传递闭包.对于任意二元同类关系 R, R^k 表示对其进行 k 次复合,即 $R^2 = R \circ R$. 则自反传递闭包由包括 R^0 在内的所有的 R 关系复合的并集构成,即:

$$R^{*k} = \bigcup_{i=0}^{k} R^i.$$

基于以上关系运算,寻址过程可定义如下:

定义 5. 寻址.

$$\mathcal{A}(\langle aco, n\rangle) = \beta(\langle aco_1, n_1\rangle, e_1) \circ \beta(\langle aco_2, n_2\rangle, e_2) \circ \cdots \circ \beta(\langle aco_k, n_k\rangle, e_k) = \beta^k.$$

对于任何一个待寻址的$\langle aco, n\rangle$来说,经过上述过程,若 e_k = null 则寻址成功执行并终止;如果 e_k = error 则表明 $\beta(\langle o_k, n_k\rangle)$没有被绑定关系定义,寻址过程因为出错而中止.当且仅当满足上述 2 个条件之一,寻址过程才真正结束.寻址过程中,k 表示穿越的 semantic scope 的数目,所有满足可达性的 aco 构成了 n 的作用域(name scope).在该作用域内,n 可被所有对象有效识别.

定义 6. 名字作用域.

$$NScope(\langle aco_x, n_x\rangle) = \{\langle aco_y, n_y\rangle \mid (\langle aco_x, n_x\rangle \mapsto \langle aco_y, n_y\rangle) \in \beta^{*k}\}.$$

2.3 名字、地址、身份标识和位置标识

基于上述概念模型,我们推导出以下通信名词的一致性形式化定义:名字(name)、地址(address)、身份标识(identifier)、位置标识(locator),并对它们之间的关系进行了阐明.

定义 7. name & address:

$\forall aco_x:\mathcal{SO}_x, aco_y:\mathcal{SO}_y, n_x, n_y$;

$\forall aco_x \mathcal{N} n_x \Rightarrow n_x =$ name of aco_x;

$\forall aco_x \mathcal{N} n_x, aco_y \mathcal{N} n_y, aco_x \mathcal{B} aco_y \Rightarrow n_y =$ address of aco_x.

如果通信对象 aco_x 与符号 n_x 之间存在命名关系,则 n_x 是 aco_x 的名字;而地址则基于绑定关系,aco_x 的地址则是另一个语义范畴内所绑定通信对象 aco_y 的名字 n_y.例如在寻址过程中,相对于 HTTP ACO 而言,URL 是名字,而解析得到的 IP 则是其地址.类似 Shoch 的定义,地址指明了“where it is”,但按照我们的定义,是基于 semantic scope 而言的相对位置.对应于 name scope (定义 6),地址的作用域称之为位置空间(location space).实际上,地址是寻址过程所引入的一种中间形式,当名字包含位置属性时,同样可被用作路由,比如基于名字的路由(route-by-name)[17].名字和地址基于命名和绑定关系,是对象的属性;而身份标识与位置标识则是针对通信中命名空间的角色而言,规定了该命名空间所承担的语义.

定义 8. Identifier & Locator:

对于在特定 name scope 内唯一的语法符号 n, $\forall \langle aco,n\rangle,\langle aco,n'\rangle \in NScope(\langle aco,n\rangle)$:

$$\exists o\mathcal{N}n, o\mathcal{N}n', aco\mathcal{N}n, aco'\mathcal{N}n \Rightarrow (n=n') \wedge (o=o'),$$

则 n 称为 aco 的身份标识; $\forall \mathcal{A}(\langle aco,n\rangle), \exists e_k=$ null,则返回的一组 n_k 为 aco 的位置标识.

在特定 name scope 内,明确、唯一标识对象的符号 n 称为身份标识. 在形式化定义中,唯一性以及对象和名字的一对一映射保证了无歧义性. 位置标识为寻址过程正常结束后(e_k=null)返回的标识,用于指明后续的转发操作. 就实际网络而言,MAC 地址和 IP 地址可分别看作是局域网和互联网通信的位置标识. 可以看出,在当前 TCP/IP 体系中,IP 地址承担了 IP ACO 位置标识和身份标识的双重角色,从而导致了移动性问题.

图 2 为当前互联网的层次式解析模型,为保证通用性,图 3 基于 semantic scope 说明了地址机制概念模型及其关系. 对通信对象而言,名字和地址是基于命名和绑定关系而产生的属性,而身份标识和位置标识则是命名空间在通信中承担的角色. 身份标识和位置标识可以产生自同一命名空间. 满足定义 8 的名字可用作身份标识,当网络对象相对于逻辑位置不发生移动,即实体与其位置属性静态绑定时,地址同样可以作为身份标识使用. 地址若要作为位置标识使用,必须在特定位置空间内能唯一标识相对位置;而包含位置属性,能在特定空间中唯一定位的名字也可以成为位置标识.

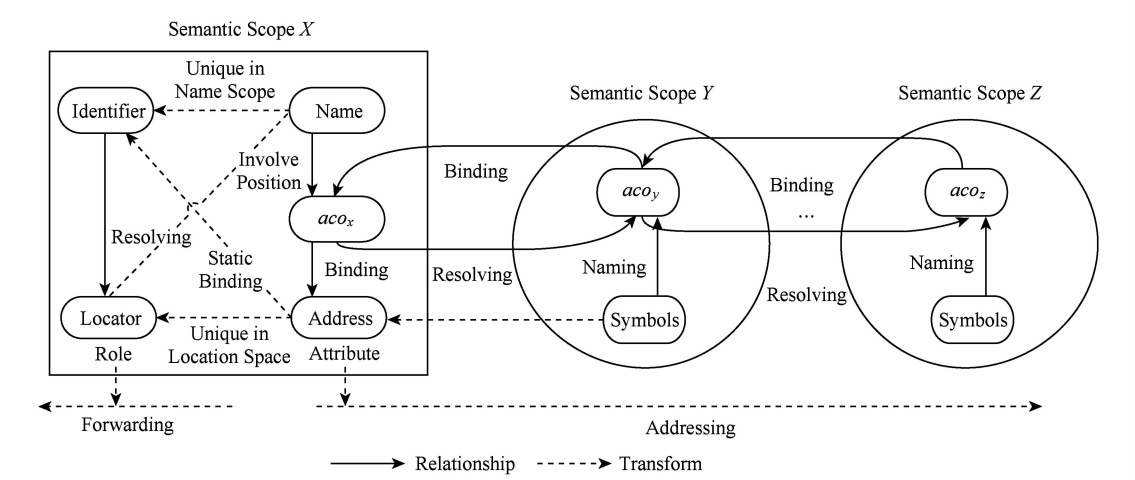


Fig. 3　The conceptual framework illustrated based on semantic scope

图 3　基于 semantic scope 的地址机制概念框架

3　地址机制的通信交互模式

上述概念模型为地址策略的设计提供了明确的解释视图以及统一的设计原则. 本节中,我们将对地址机制中的动态交互过程进行多维度抽象,获取核心的标准接口以及交互模式,以构造一种可以兼容异构地址机制的最小化通用过程处理框架. 在进行地址机制设计时,该框架能够依据交互模式以及行为语义对通信行为进行正确性检查.

3.1　交互模式与原语

根据抽象通信模型的规约描述,从消息传递的角度来看,抽象通信对象 ACO 的外部行为由接口行为和交换行为共同构成. 接下来我们将介绍这些行为的具体操作语义,同时也作为交互过程正确性的证明基础. 通信顺序进程(communicating sequential processes, CSP)[18]是一种适合描述并发和通信系统的常用形式化描述技术,其描述可导入模型检验器进行相关性质的验证. 我们选用 CSP 相关符号来刻画交互行为语义. CSP 的基本单元是进程(process),进程刻画了通信对象的行为模式,对象行为集合构成了该进程的事件集 αP. "$a\rightarrow P$"表示先执行事件 a 然后进程 P,$\surd$ 表示 1 个成功执行的事件. 为方面描述,定义 $\delta=\surd\rightarrow$STOP,代表该进程成功终止(类似于 CSP 中的 SKIP). $P \parallel Q$ 为进程 P 和 Q 的并发进程,$P \sqcap Q$ 代表行为为 P 或者 Q 的行为的进程,内部选择算子$\sqcap$表示该进程能够自主选择进程. $P \sqcap Q$ 表示该复合进程行为依据第 1 个事件来决定的进程,选择复合算子$\sqcap$说明该进程的选择只能由外部环境触发.

接口域(I/F)定义了转发操作,包括一组成对出现的事件,如 send/receive,invoke/return 等. 交换服务 SP 描述了本地或者全局协作的交换行为,由定义 4 可知,交换行为的目的是为了寻址. 对象 aco 的操作行为语义可以形式加以规约:

$aco=\mathrm{I/F}\parallel \mathrm{SP}$, where $_{\alpha}aco=_{\alpha}\mathrm{I/F}\cup_{\alpha}\mathrm{SP}$.

当上述过程完成之后,适配服务对转发的消息进行适配处理,并输出到实例化的通道上. 根据以上描述,对于特定对象,完成的通信过程($COMM$)可定表述为 3 种抽象交互模式的组合:寻址(addressing)、转发(forwarding)、适配(adaption). 采用 CSP 的并发进程形式定义为

$COMM=\mathrm{I/F}(m)\parallel \mathrm{SP}(\langle aco,fi\rangle)\parallel \mathrm{AP}(m)$.

基于上述观察,我们对每一种交互模式分别定义标准的接口(事件表),这些原语涵盖了整个转发、寻址的抽象过程,可以灵活构造地址机制中所期望的交互行为. 所有的通信对象支持相同的接口组,具体实现机制可以根据实际通信场景灵活设计. 消息类型 $message$ 以及 fi 分别抽象引用了通信分组以及用于转发决策的头部域,在具体的实现中,将被替换为具体协议相关的数据结构,如身份标识、位置标识等. 另外,对象类型 aco 同样可以被继承,扩展成为实际的通信主体.

1) 转发模式

$fi\ Getfi\ (msg)$:

从消息中获取相应转发指令 fi 并返回,具体的转发信息可依据具体机制实例化.

$send\ (aco,msg)\Leftrightarrow msg!A^{\mathrm{SOP}}\rightarrow msg!C^{\mathrm{SOP}}$:

对应于通信性质 1,将消息发送到对端的输入端口,值得注意的是,从逻辑上来说,该消息还没有进行适配处理.

$\langle aco,msg\rangle receive()$:

接收消息并返回该消息和发送方 aco.

$message\ copy\ (msg)\Leftrightarrow msg?A^{\mathrm{SIP}}\rightarrow msg!A^{\mathrm{SOP}}$:

将消息拷贝到输出端口.

2) 寻址模式

$message\ request(\langle aco,fi\rangle)$:

将请求解析的 fi 放入请求消息中.

$\{\langle aco,fi\rangle\}\ lookup\ (\langle aco,fi\rangle)$:

该原语执行解析操作,映射系统的具体实现机制可以灵活定义,不在本文讨论的范围内. 由性质 2 可知,该原语返回一组$\langle aco,fi\rangle$.

$message\ response(fi,aco)$:

将查询的结果放在响应消息中.

3) 适配模式

$encap(fi,msg)$:

为消息封装 1 个协议相关的头部,通常用于隧道的入口处或协议层之间.

$fi\ decap(fi,msg)$:

解封装,从消息中移除新增的头部前缀. 通常用于隧道的出口处.

$encrypt(method,msg)$:

使用加密机制对消息进行加密,具体的加密方法可以灵活定义.

$swap(fi,msg)$:

对消息中转发指令 fi 进行替换,例如 MPLS 和 NAT 中的标签交换.

$deliver(msg)$:

消息不经过任何处理,直接转发.

3.2 通道实例及语义规约

在图 1 的通信模型中,适配服务作为通道行为的一部分提供,目的在于分离对象的转发行为与对象间的交互行为,实现通道实例的可重用,从而降低通信过程的复杂性. 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. 通道实例基于角色(role)和粘合剂(glue)进行语义说明. role 描述了 aco 所期望的外部行为规范;glue 用于规定各个 role 的行为时序,协调抽象通信对象间的交互行为. 基于适配模式对消息的操作,我们定义通道实例:

Channel $Encap\text{-}Channel=$

role $sender=(encap(fi,msg)\rightarrow send(aco,msg)\rightarrow\surd)$;

role $receiver=(receive(aco,msg)\rightarrow\surd)$;

glue $Eglue=(encap(fi,msg)\rightarrow send(aco,msg)\rightarrow receive(aco,msg)\rightarrow\surd)$.

该通道实例对消息进行封装操作,可连接不同协议层实现通信. 考虑消息从传输层加上 IP 头部,再发送给网络层接收.

Channel $Tunnel=$

role $peer=(encap(fi,msg)\rightarrow send(aco,msg)\rightarrow\surd)$;

role $peer=(decap(fi,msg)\rightarrow receive(aco,msg)\rightarrow\surd)$;

glue $Tglue=(encap(fi,msg)\rightarrow send(aco,msg)\rightarrow decap(fi,msg)\rightarrow receive(aco,msg)\rightarrow\surd)$.

通道的隧道实例,用于对等层通信. 例如 IPv4/IPv6 基于隧道的过渡方式,fi 被实例化为 IPv6 头

部,边缘路由器使用 v6 包头对分组封装,对端边缘路由器进行解封装并接收消息.

Channel *Gateway-Channel*=

role *sender*=(*swap*(*fi*, *msg*)→*send*(*aco*, *msg*)→√);

role *receiver*=(*receive*(*aco*,*msg*)→√);

glue *Gglue*=(*swap*(*fi*, *msg*)→*send*(*aco*, *msg*)→*receive*(*aco*,*msg*)→√).

通道的网关模型.该模型与封装方式的区别在于替换而不是封装标签,例如 NAT 以及 MPLS 中的标签交换.

Channel *C/S-Channel*=

role *client*=(*request*(⟨*aco*, *fi*⟩)→*receive*())⊓ð;

role *server*=(*lookup*(⟨*aco*, *fi*⟩)→*response*(*msg*, *aco*))⊓ð;

glue *Cglue*=(*request*(⟨*aco*, *fi*⟩)→*lookup*(⟨*aco*,*fi*⟩)→*response*(*msg*,*aco*)→*receive*())⊓ð.

客户服务器交互通道,适用于解析请求或者过程调用等通信连接.注意客户角色使用内部选择符而服务器角色使用的是外部选择符,也就是说,客户端能够主动结束请求,而服务器不能主动结束服务,只能由外部触发决定其结束与否.基于该通道设计客户服务器模型可以根据通道语义得到有效验证.

另外,我们还定义了加密通道、转发通道等其他实例,这里不再一一赘述.

该通用过程框架包含了经过观察、精心选择和构造的核心接口原语,可快速灵活地构建交互模式实例,通信对象之间通过通道实例交互,实现转发与交互行为的松耦合.

4　构造与验证实例

为论证上述概念模型与交互过程框架该的完备性和正确性,我们以 ID/Locator 地址分离体系中的映射-封装机制(map-encap)[9]为例,来说明一种地址方案如何在该框架内表述和构造并得到有效验证.

如图 4 所示,map-encap 机制通过网络提供功能以实现位置与身份相分离,身份标识 EID 只在接入网中路由,隧道边缘路由器(ingress tunnel router, ITR)通过查询映射系统(resolver)获取全局路由标识 RLOC,并将其封装在消息头部,再对端边缘路由器 ETR 处解封装.我们假定 EID 和 RLOC 分别来自 IPv4 以及 IPv6 地址空间,类似于 IPv6 的隧道过渡机制.为保证静态模型与交互过程的相对独立,对应于概念模型和过程框架,下面我们从配置定义和交互过程定义 2 个方面论证 map-encap 地址机制的实现.

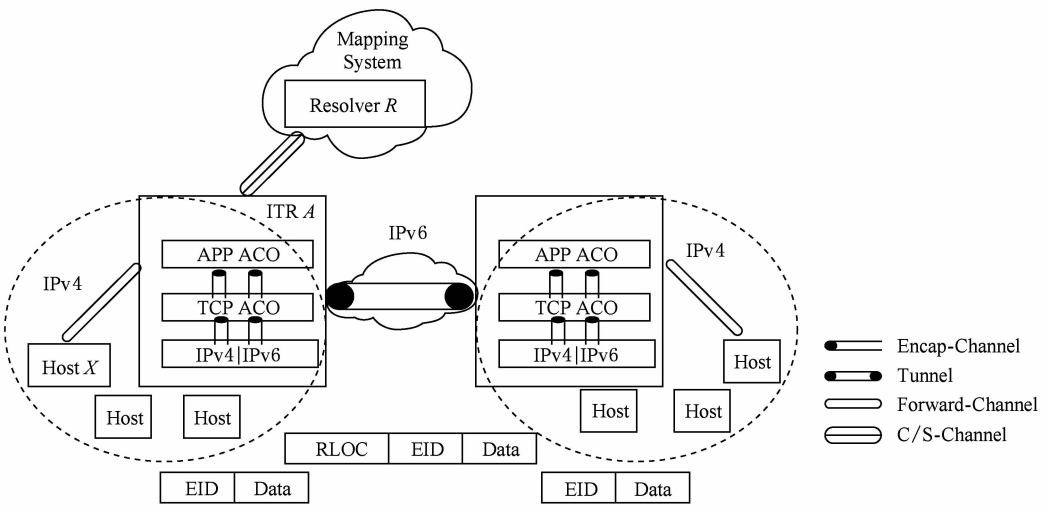


Fig. 4　Abstract processing of map-encap scheme

图 4　Map-encap 地址机制抽象过程

4.1　配置定义

考虑到配置描述的精确性和可重用性,我们基于 XML 对地址方案进行描述,并提出了 3 类 XML 描述规范(schema):通信实体定义、拓扑配置以及命名空间语法语义规则.描述规范严格对应于本文提出的概念模型,以保证一致性的设计约束规范检查.图 5 为 map-encap 地址机制的 XML 描述,限于篇幅,我们截取部分定义,并省略过多的 XML 语法符号.

通信实体配置定义了对象的继承或聚合机制;

拓扑描述实体基于通道实例的连接关系,同时包含了交互行为的语义约束;命名空间包括语义规范和语法规则 2 个方面. 本文第 2 节定义的概念约束了语义规范的设计,并能够进行检查;语法规则规定了数据类型,比如以正则表达式匹配地址格式等. 基于上述 3 类 XML schema,地址机制可被灵活定义描述,同时保证设计方案的一致性和正确性.

```
〈!-- Entity Configuration -- 〉
〈Entity name="A" type="ITR"〉
    〈ACOs〉APP, TCP, IPv6, IPv4〈/ACO〉
    〈Assemble〉Encap-Channel〈/Assemble〉
〈/Entity〉
〈!-- Topology Configuration-- 〉
〈Topology〉
  〈Channel type="Tunnel" from="A" to="B" /〉
  〈Channel type="C/S-Channel" from="A" to="R"/〉
〈/Topology〉
〈! -- Namespace -- 〉
〈Namespace name="IPv6" role="Locator"〉
  〈!-- Semantic --〉
  〈Semantic scope〉 IPv6 ACO 〈/Semantic scope〉
  〈Namescope〉 APP, TCP, IPv6, IPv4 〈/Namescope〉
  〈binding〉 IPv4 namespace 〈/binding〉
  〈!-- Syntax--〉
  〈xs: restriction base="HexBinary"〉
  〈xs: pattern value="([A-Fa-f0-9]{1,4}:){7}[A-Fa-f0-9]
    {1,4}"/〉
  〈/xs: restriction〉
〈/Namespace〉
```

Fig. 5 A snippet of configuration description

图 5 部分配置描述

4.2 交互过程

在 3.1 节中,我们提出了一组精炼的核心接口原语,为验证其通用性,我们以图 4 中 ITR A 的通信行为为例,基于上述原语来构建 3 种基本交互模式,从而实现 map-encap 地址机制抽象通信过程. 另外,我们基于断言技术,用一组谓词公式来刻画程序在执行过程中的状态,通过考察各断言能否成立,实现对程序正确性的证明;图 6 以伪代码形式刻画了 ITR A 的一般抽象通信行为,并插入了适当的断言.

可以看出,基于标准接口构建的 3 种基本交互模式完成了 ITR A 的通信行为,而诸如 lookup 等接口则依赖于具体协议机制实现. 在实现中,并不需要所有的抽象对象都具有完备的接口实现,这里描述的是通用的交互过程,任何实际策略均可通过与具体协议相关的继承、改写、实例化而获得.

```
Process (aco pre, message msg) {
pre-condition
{φ: pre=X∧(fi!X^SOP→fi?A^SIP)}
  /* forwarding */
  String n=getfi(fi, msg);
  request(〈pre,n〉);
  send(〈pre, n〉, R);
  }
  {〈aco, fi〉}A; /* addressing results */
  for e in binding relation;
  /* addressing */
  while (e≠error) do {
    {σ1: fi!R^SIP∧(fi=n)∧A=∅}
      if (e≠null) {
        {〈pre,n′〉}A=lookup(〈pre, n〉);
        } else {
          response(〈pre, n′〉);
          send(A, response);
        }
    {σ2: fi?A^SIP∧(fi=n′)∧A≠∅}
      /* adaption */
      receive(R, response);
      encap(n′, msg′);
      Copy(msg′);
    post-condition
    {ψ: (fi!A^SOP→ fi?A^SIP)∧(fi=n′)∧(msg=msg′)}
}
```

Fig. 6 The abstract processing

图 6 抽象通信过程

4.3 程序正确性证明

程序正确性证明指的是通过形式化的推导,评价一个程序是否符合特定规范. 本节我们使用 Hoare 逻辑,以 1.1 节以及 3.1 节中的通信性质和行为语义为证明依据,来证明上述抽象过程的正确性. 文献[19]针对转发过程中名字和地址的移除模式,利用 Hoare 逻辑证明了转发过程的正确性. 图 7 为 Hoare 逻辑的规则描述.

Assignment	$\overline{\{\varphi[e/x]\}x:=e\{\varphi\}}$
Sequential	$\dfrac{\{\varphi\}Q_1\{\sigma\},\{\sigma\}Q_2\{\psi\}}{\{\varphi\}Q_1;Q_2\{\psi\}}$
IfThenElse	$\dfrac{\{\varphi\wedge B\}Q_1\{\psi\},\{\varphi\wedge\neg B\}Q_2\{\psi\}}{\{\varphi\}\text{if } B \text{ then } Q_1 \text{ else } Q_2\{\psi\}}$
ForEach	$\dfrac{\{\varphi\wedge B\}Q\{\psi\}}{\{\varphi\} \text{ while } B \text{ do } Q\{\psi\wedge\neg B\}}$
Consquence	$\dfrac{\varphi\rightarrow B,\{B\}Q\{\psi\}}{\{\varphi\}Q\{\psi\}}$

Fig. 7 Hoare logic rules

图 7 Hoare 逻辑的规则描述

Hoare 逻辑[20]是对断言方法的推广,并建立了 1 套公理系统,其公理和命题的一般形式为:$\{\varphi\}\ Q\ \{\psi\}$. 其中 φ,ψ 为逻辑表达式,Q 是 1 个程序段. 整个

表达式的含义是:如果在 Q 执行之前有 φ 成立,并且 Q 能终止执行,则必有 ψ 成立,通常将 φ,ψ 称为前置断言(pre-condition)和后置断言(post-condition).为了证明不变式语句,Hoare 公理系统引入了 1 条赋值公理和 4 种推理规则,如图 7 所示,其中横线上方的语句为假设,下方是可推导出的结论,其语义为:如果所有的假设成立,则结论必然成立.对于 map-encap 机制而言,合理的前置断言应为主机 X 将需要解析的 fi 发送到 A 的输入端口;当一系列交互过程完成后,后置断言为:A 将含有解析完毕,经过封装的 fi' 拷贝到输出端口,基于上述规则,我们在程序以及循环的开始和结束处分别插入断言.

$$\varphi_i\ pre=X\wedge(fi!X^{\mathrm{SOP}}\rightarrow fi?A^{\mathrm{SIP}});$$

$$\sigma_{1i}\ fi!R^{\mathrm{SIP}}\wedge(fi=n)\wedge(\mathcal{A}=\varnothing);$$

$$\sigma_{2i}\ fi?A^{\mathrm{SIP}}\wedge(fi=n')\wedge(\mathcal{A}\neq\varnothing);$$

$$\psi_i\ (fi!A^{\mathrm{SOP}}\rightarrow fi?A^{\mathrm{SIP}})\wedge(fi=n')\wedge(msg=msg').$$

下面对证明过程进行简要说明:为方便描述,我们将断言之间的程序段分别以其交互模式 forwarding,addressing,adaption 表示,其最终证明形式可表示为

Goal 1:$\{\varphi\}$process$\{\psi\}$,由顺序规则可转换为 3 个 Hoare 三元组的证明:

$$\text{Goal 2}:\{\varphi\}\ \text{forwarding}\ \{\sigma_1\};$$

$$\text{Goal 3}:\{\sigma_1\}\ \text{addressing}\ \{\sigma_2\};$$

$$\text{Goal 4}:\{\sigma_2\}\ \text{adaption}\ \{\psi\}.$$

1) 为证明 Goal 2,即转发后消息到达 R 并且寻址没有完成,由赋值公理和推断规则,只需证明:

$$[\{\varphi\}\wedge send()]\Rightarrow\{\sigma_1\}\Leftrightarrow(fi!X^{\mathrm{SOP}}\rightarrow fi?A^{\mathrm{SIP}})\wedge(fi!A^{\mathrm{SOP}}\rightarrow fi?R^{\mathrm{SIP}})\Rightarrow pre=X\wedge(fi!X^{\mathrm{SOP}}\rightarrow fi?A^{\mathrm{SIP}}).$$

由通信的传递性(性质 3)可知,上述推断成立即 Goal 2 得证.

2) 为证明 Goal 3,根据循环规则,只需证明:$\{\sigma_1\wedge(e\neq\text{error})\}$ if $(e\neq\text{null})$ then *lookup* else *send* $\{\sigma_2\}$,再由条件规则转化为

Goal 5:$\{fi!R^{\mathrm{SIP}}\wedge(fi=n)\wedge(\mathcal{A}=\varnothing)\wedge(e\neq\text{error})\wedge(e\neq\text{null})\}$ *lookup*()$\{fi?R^{\mathrm{SIP}}\wedge(fi=n)\wedge(\mathcal{A}=\varnothing)\}$;

Goal 6:$\{fi!R^{\mathrm{SIP}}\wedge(fi=n)\wedge(\mathcal{A}=\varnothing)\wedge(e\neq\text{error})\wedge(e=\text{null})\}$ *send*()$\{fi?A^{\mathrm{SIP}}\wedge(fi=n')\wedge(\mathcal{A}\neq\varnothing)\}$;

Goal 5 的含义为寻址过程没有结束,对其使用赋值公理、推断规则、交换性质(性质 2)以及寻址定义(定义 6),Goal 5 得证,即:

$$fi!R^{\mathrm{SIP}}\wedge(fi=n)\wedge(\mathcal{A}=\varnothing)\wedge(e\neq\text{error})\wedge(e\neq\text{null})\ SP_R(\langle A,fi\rangle)\Rightarrow fi?R^{\mathrm{SIP}}\wedge(fi=n)\wedge(\mathcal{A}=\varnothing).$$

同理,Goal 6 的含义为寻址结束并将结果返回 A,由通信传递性可以得到证明,即:

$$fi!R^{\mathrm{SIP}}\wedge(fi=n)\wedge(\mathcal{A}=\varnothing)\wedge(e\neq\text{error})\wedge(e=\text{null})\wedge(fi!A^{\mathrm{SOP}}\rightarrow fi?R^{\mathrm{SIP}})\Rightarrow fi?A^{\mathrm{SIP}}\wedge(fi=n')\wedge(\mathcal{A}\neq\varnothing).$$

综上,Goal 3 得到了证明.

3) 为证明 Goal 4,即 A 接收到解析结果并封装在新消息中,发送到输出端口.由赋值公理、推断规则以及 *encap*,*copy* 的行为语义可得:

$$fi?A^{\mathrm{SIP}}\wedge(fi=n')\wedge(\mathcal{A}\neq\varnothing)\wedge encap\wedge copy\Rightarrow fi?A^{\mathrm{SIP}}\wedge(fi=n')\wedge(\mathcal{A}\neq\varnothing)\wedge msg'\wedge(msg!A^{\mathrm{SIP}}\rightarrow msg'?A^{\mathrm{SOP}})\Rightarrow(fi!A^{\mathrm{SOP}}\rightarrow fi?A^{\mathrm{SIP}})\wedge(fi=n')\wedge(msg=msg').$$

Goal 4 得证.　　证毕.

综上,Goal 1 即上述程序描述的正确性到了证明.

5　通用地址机制引擎

当前互联网体系结构只允许对顶层的应用和技术创新,而作为核心的地址组件却难以扩展,组件之间的紧耦合特性阻碍了体系结构的进一步演进,这也是当前互联网陷入僵化的主要原因之一.为增加地址标识机制的灵活性和通用性,基于本文提出的概念与过程交互模型,我们实现了通用地址机制引擎(universal engine of address schemes, UEAS)原型系统,旨在提供一种通用平台以容纳或兼容互联网地址类型的异构性和多样性.任何所期望的地址机制都可以在一致性和正确性的约束下,快速、灵活地构造,并针对特定策略的性能开销进行有效评估.

如图 8 所示,通用地址机制引擎的结构框架.网络通信实体、命名空间语法语义和拓扑配置以 XML 描述并存储在本体库中,以模板形式调用.最核心的抽象基类是本文一直强调的 ACO,Message 以及 Channel,在引擎中以 Java 抽象类定义,并提供了基本的操作原语.具体通信主体可以通过对 3 种抽象类的继承或实例化而获取.3 种基本交互模式以及相应的操作原语以类库形式提供,供通信主体调用以构造实际通信过程.为了进一步增加功能的完备性,我们提供了 1 个组件库供设计时以服务形式调用,包括安全机制、前缀聚合机制、地址映射机制等.

我们为组件库的每个服务组件指定了 1 个标识符，应用调用的时候只需传递这个标识符，实现了组件库的可扩展性而无需影响已有应用. 引擎管理器解析 XML 配置文件以及定义交互模式，并根据本文前面介绍的概念模型与组件语义进行正确性、完备性检查，最后生成地址机制实例运行.

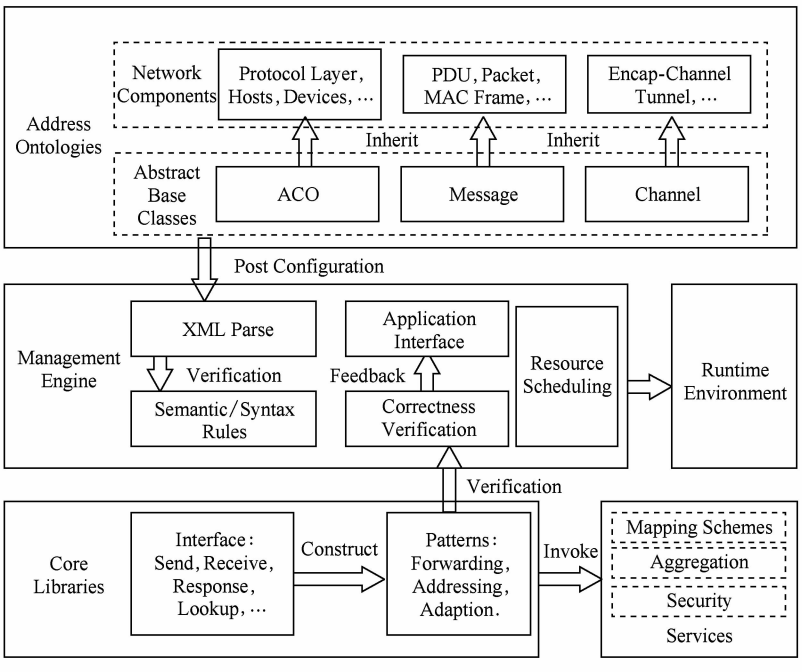


Fig. 8 Universal engine of address schemes

图 8 通用地址机制引擎

通用地址框架可为异构的地址标识方案提供统一的构造、部署平台. 研究人员调用或者自定义本体库中的通信组件，基于我们预先定义的 XML 描述规范，将特定地址机制形成脚本描述并提交至引擎管理器；引擎管理器依据语法语义规则对该脚本进行验证和解析，并调用相应服务组件以实例化地址机制并注入运行时网络.

另外，通用地址机制框架并非列举地址体系中所有的通信组件或行为，而是通过精炼 1 个最小化静态核心来保证设计的完备性和最大限度的自由性. 比如框架内并没有限定地址的聚合机制以及映射更新机制等，而是允许设计人员根据实际地址策略灵活定义、调整以及评估，从而选择其中最优的设计方案.

6 结束语

本文通过对互联网地址体系异构及多样性的深入研究，抽象出其本质特征属性，提出了完备的形式化概念模型，并利用精心构造的最小化交互模式来灵活构造地址机制的抽象通信过程. 基于上述概念与过程框架，推导出可兼容异构地址策略并存的通用平台以及其原型系统，任何期望的地址机制都可以在概念及行为语义的约束下灵活定义实现.

互联网地址机制中的基本概念存在歧义以及不一致性，已经在学术界争论了几十年之久. 在研究与命名寻址相关的方法论之前，重新审视这些关键概念是具有重要意义. 本文以一种形式化的概念表述，赋予这些术语精确而又一致的定义，同时包含了相关的设计原则约束，在实际的地址机制设计中具有一定的指导作用. 而对于动态的过程交互仅仅是通过定义一组核心标准接口构造 3 种基本模式来实现，最大限度保证了设计的自由性. 结合本文推导出的通信性质，设计的正确性同样可以得到有效的形式化验证. 最后，通用地址引擎为进一步探索互联网地址机制的演进方向提供了可能的实验平台. 未来工作中，我们将结合资源调度，基于对应用的支撑能力，提出相应的评估机制，并对原型系统进行完善，使其能够获取量化的评估结果.

参 考 文 献

[1] Zheng Qinghua. The introduction of application technology-oriented "Internet+"[J]. Journal of Computer Research and Development, 2015, 52(12): 2657-2658 (in Chinese)

(郑庆华. 面向“互联网+”的应用技术专题前言[J]. 计算机研究与发展, 2015, 52(12): 2657-2658)

[2] Xu Ke, Zhu Min, Shen Meng. The application-oriented evolvable Internet architecture [J]. Communications of the CCF, 2015, 11(6): 37-42 (in Chinese)
(徐恪, 朱敏, 沈蒙. 面向应用的可演进互联网体系结构[J]. 中国计算机学会通讯, 2015, 11(6): 37-42)

[3] Xu Ke, Zhu Min, Lin Chuang. Internet architecture evaluation models mechanisms and methods [J]. Chinese Journal of Computers, 2012, 35(10): 1985-2006 (in Chinese)
(徐恪, 朱敏, 林闯. 互联网体系结构评估模型、机制及方法研究综述[J]. 计算机学报, 2012, 35(10): 1985-2006)

[4] Zhu Min, Xu Ke, Lin Song. The evaluation method towards the application adaptability of Internet architecture [J]. Chinese Journal of Computers, 2013, 36(9): 1785-1798 (in Chinese)
(朱敏, 徐恪, 林嵩. 面向应用适应能力的互联网体系结构评估方法[J]. 计算机学报, 2013, 36(9): 1785-1798)

[5] Ahlgren B, Brunner M, Eggert L. Invariants: A new design methodology for network architectures [C] //Proc of ACM SIGCOMM'04 Workshop on Future Directions in Network Architecture. New York: ACM, 2004: 65-70

[6] Rexford J, Dovrolis C. Future Internet architecture: Clean-slate versus evolutionary research [J]. Communications of the ACM, 2010, 53(9): 36-40

[7] Feldmann A, Telekom D, Berlin L T. Internet clean-slate design: What and why? [J]. ACM SIGCOMM Computer Communication Review, 2013, 37(3): 59-64

[8] Xu Ke, Zhu Liang, Zhu Min. Architecture and key technologies of Internet address security [J]. Journal of Software, 2014, 25(1): 78-97 (inChinese)
(徐恪, 朱亮, 朱敏. 互联网地址安全体系与关键技术[J]. 软件学报, 2014, 25(1): 78-97)

[9] Ramirez W, Masip-Bruin X, Yannuzzi M. A survey and taxonomy of ID/Locator split architectures [J]. Computer Networks, 2014, 60(2): 13-33

[10] Andersen D G, Balakrishnan H, Feamster N, et al. Accountable Internet protocol (AIP) [C] //Proc of ACM SIGCOMM'08. New York: ACM, 2008: 339-350

[11] Clark D, Braden R, Falk A, et al. FARA: Reorganizing the addressing architecture [C] //Proc of ACM SIGCOMM 2003. New York: ACM, 2003: 313-321

[12] Atkinson R, Bhatti S, Hailes S. ILNP: Mobility, multi-homing, localised addressing and security through naming [J]. Telecommunication Systems, 2009, 42(4): 273-291

[13] Balakrishnan H, Lakshminarayanan K, Ratnasamy S, et al. A layered naming architecture for the Internet [C] //Proc of ACM SIGCOMM 2004. New York: ACM, 2004: 343-352

[14] Choi J, Park C, Jung H, et al. Addressing in future Internet: Problems, issues, and approaches [C] //Proc of the 3rd Int Conf on Future Internet Technologies. Piscataway, NJ: IEEE, 2008: 123-129

[15] Shoch J F. Internetworking naming, addressing, and routing [C] //Proc of the 17th IEEE Computer Conf. Piscataway, NJ: IEEE, 1978: 72-79

[16] Saltzer J. On the Naming and Binding of Network Destinations: RFC1498 [S/OL]. Fremont, CA: IETF, 1993 [2015-11-19]. http://www.rfc-editor.org/rfc/rfc1498.txt

[17] Zhang L, Afanasyev A, Burke J, et al. Named data networking [C] //Proc of ACM SIGCOMM 2014. New York: ACM, 2014: 66-73

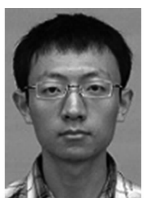
[18] Hearer C A R. Communicating sequential processes [J]. Communications of the ACM, 1978, 21(1): 666-677

[19] Karsten M, Keshav S, Prasad S, et al. An axiomatic basis for communication [C] //Proc of ACM SIGCOMM 2007. New York: ACM, 2007: 217-228

[20] Hoare C A R. An axiomatic basis for computer programming [J]. Communications of the ACM, 1969, 12(10): 576-580

Zhu Liang, born in 1982. PhD candidate. His main research interests include Internet architecture and its evaluation.

Xu Ke, born in 1974. Professor, PhD supervisor. His main research interest include Internet architecture, high performance router, P2P network, Internet of things and network economics.



Xu Lei, born in 1983. PhD candidate. His main research interests include datacenter networking and software-defined networking.