一种虚拟化深度包检测部署机制

汪学舜 余少华 戴锦友
(武汉邮电科学研究院 武汉 430074)

摘 要 网络功能虚拟化转变了网络架构和网络业务的部署。在网络功能虚拟化架构中,实现虚拟化深度包检测只需在传输路径上进行一次扫描,但高效部署深度包检测功能引擎成为难题。将深度包检测功能部署问题形式化为线性规划问题以满足约束条件,并提出一种基于代价最小的贪婪算法和优化的贪婪算法来解决深度包检测功能部署问题。该算法对部署代价和网络资源代价进行折衷,实现了最小化的部署代价。实验结果表明,所提算法能够实现深度包检测功能部署并取得近似最优解。

关键词 深度包检测,部署,网络功能虚拟化,最小代价优化

中图法分类号 TP393 **文献标识码** A **DOI** 10.11896/j.issn.1002-137X.2017.08.017

Virtualization Deep Packet Inspection Deployment Method

WANG Xue-shun YU Shao-hua DAI Jin-you
(Wuhan Research Institute of Post and Telecommunication, Wuhan 430074, China)

Abstract Network function virtualization (NFV) changes the network architecture and the deployment of network services. Traffic is scanned only once in the virtual network architecture for the virtualization deep packet inspection (DPI), but DPI deployment is a difficult problem. In this paper, DPI engine deployment was formulated as linear programming problem (ILP) to satisfy some constraints. A greedy algorithm based on cost minimization and an optimal greedy algorithm were proposed to solve the function depoyment problem of deep paclet inspection. The proposed algorithm compromises the DPI deployment cost and network resource cost, and minimizes the cost of deployment. Simulation results show that the proposed scheme can achieve the approximate optimal solution of DPI deployment.

Keywords Deep packet inspection, Deployment, Network function virtualization, Cost minimization

1 引言

深度包检测(Deep Packet Inspection,DPI)通过流量分析设备部署在网络中。网络功能虚拟化(Network Function Virtualization,NFV)的逐步实施以及软件定义网络(Software Defined Networks,SDN)中灵活的路由能力,使得深度包检测的部署发生了变化:对深度包检测功能进行虚拟化,将其作为一个综合软件功能部署在服务器上。

文献[1]描述了当今网络传输中数据流在到达目的之前需要通过一系列中间设备进行DPI处理,这意味着数据流被中间设备DPI组件进行一遍又一遍的扫描,如图1所示,中间设备独立执行DPI操作;而网络发展趋势是将中间设备DPI处理虚拟化作为一个业务,只需要在一个位置进行DPI操作[2],其他设备共享使用,如图2所示。

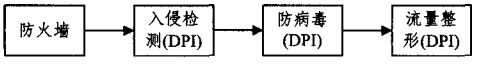

图1 传统网络DPI实现示意图

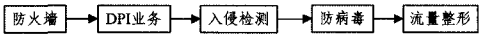

图2 网络功能虚拟化DPI实现示意图

虚拟化DPI的实现面临着一系列挑战,其中最主要的问题是网络中DPI功能的快速部署。在部署过程中需要考虑以下问题:1)对所有流进行监控,意味着可能需要在所有网络节点部署虚拟深度包检测功能。为了降低部署代价,需要使用尽量少的DPI实例,并对网络流量进行重路由。2)通过监控确定部署DPI实例的数量以及部署节点。

本文对DPI部署问题进行形式化,对任意给定的部署方式和相关的流,定义目标函数进行评估。目标函数包含多种不同的目标因素,虚拟DPI部署是这些目标因素的折衷。本文提出集中式代价最小贪婪算法及其改进算法,实现了确定DPI部署的数量和位置,所提算法可使代价降到最低,也可用于自适应动态深度包检测。最后本文通过实验评估其有效性。

2 相关工作

网络功能虚拟化得到网络运营商的广泛支持,国际互联

到稿日期:2016-06-12 返修日期:2016-09-28 本文受863项目:未来一体化标识网络关键技术和示范(2015AA015702)资助。

汪学舜(1970—),男,博士,高级工程师,主要研究方向为流媒体数据传输QoS保证、网络虚拟化和深度包检测、大数据网络等,E-mail:wang_xueshun@163.com;**余少华**(1963—),男,博士,教授,主要研究方向为IP互联网、深度包检测、光通信和光传输网络;**戴锦友**(1968—),男,博士,主要研究方向为深度包检测、大数据网络。

网标准组织 IETF 和 ITU 成立了几个研究组对 NFV 展开研究[3]，如 IETF 标准组的 RFC7498 和 RFC7665 均是一种 NFV 的实现。NFV 包括将网络传输功能作为软件虚拟实例，如防火墙、存储、计算、负载均衡等网络功能都可以通过虚拟化实现。由于深度包检测的传输模式匹配和动态特性更易实现网络功能虚拟化，因此 DPI 设备供应商纷纷支持虚拟化，并提供 DPI 软件实现[4]。将多个硬件实现的功能进行虚拟化后，在服务器上安装相应的虚拟化功能软件即可实现 DPI 功能，使得业务可以快速部署。

对于 NFV，很多研究者只关注数据中心虚拟机的部署，并没有从整体上考虑计算、存储和网络的实施，如文献[5]从负载均衡或者节能方面对服务器部署进行了优化，但数据中心中虚拟机的部署优化与整体网络功能虚拟化的优化存在区别，二者优化的设备种类存在较大差别。

在 DPI 设备虚拟化方面，文献[6]提出网络设备的部署机制，在虚拟环境中防护网络攻击，VM 可以在不同机器设备上进行迁移；受 SDN 启发，文献[7]对网络设备进行标准化控制；为了减少设备操作代价高的中间设备，文献[8]将网络设备外部功能作为一个业务，并将其作为网络的外部实体进行管理。

在虚拟化网络设备部署方面，文献[9]研究了软件网络交换机的性能，文献[10]对硬件实现 DPI 进行了研究。但很少有研究者对虚拟网络功能的部署优化进行研究，文献[11]定义了一种虚拟网络功能业务链的描述语言，可用于描述分布式网络中业务链的部署问题；文献[12]描述了基于启发式算法的虚拟深度包检测部署；文献[13]对传统网络的部署问题进行监控，但并不适用于较大的网络。

3　深度包检测部署机制与模型

文献[2]提出将不同中间设备的 DPI 引擎提取出来，并将其作为网络中的一个业务，该业务通过部署 1 个或多个业务实例进行实现，由逻辑中心的 DPI 控制器进行控制。在 NFV 中，DPI 引擎部署通过虚拟化 DPI 实例进行。

本节定义了 DPI 部署问题，并提出了线性规划算法。

3.1　模型和目标定义

DPI 部署问题可形式化如下：对于给定的 NFV 架构和给定的业务传输需求，查找一个 DPI 引擎的部署，使得总代价最小。该代价最小优化问题是以下目标最小化的结合：1）部署 DPI 引擎代价；2）网络中进行流重定向代价；3）不同的操作约束。这些代价包含部署 DPI 引擎的经济代价，如 license 价格、CPU 利用率、能量消耗等，以及网络资源的代价，如网络运营商总体收入、网络接纳新业务的能力等。受到的约束主要包括管理限制，如最大部署引擎数量限制、每条链路最大带宽限制、不受监控流的最大数量等。

最小化 DPI 部署代价有以下两个主要目标：1）DPI 引擎数量最少；2）网络负载最小。这两个目标是强相关的。由于要求所有流至少经过一个 DPI 引擎，以便对业务进行分析，当 DPI 引擎数量较少时，为使所有数据流都能处理，需要延长网络传输的路径，因此较少的 DPI 引擎数量会增加网络带宽负载。另一方面，增加部署 DPI 引擎数量可使带宽利用率降低，节省网络运行成本，但增加了部署 DPI 引擎的成本。图 3 示出的 DPI 引擎部署说明了 NFV 架构下 DPI 引擎数量与带宽利用率之间的相关性，DPI 引擎数量减少，如图 3 中示例为 1 个（Node E），导致数据流（黑色粗线所示）被重定向，则增加网络带宽负载；反之，部署多个 DPI 引擎，使每条数据流最短路径传输，则网络负载减小。

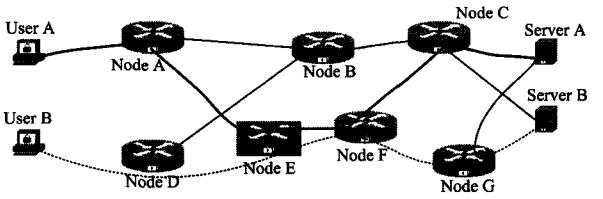

图 3　DPI 引擎部署示意图

3.2　线性规划形式化

网络模型可定义为一个无向连接图 $G(V,E)$，其中 V 为网络节点集合，E 为网络节点之间物理链路组成的边集合，每一条边的传输能力为 C_{ij}。每一个节点表示 NFV 架构中的网络节点，都具有部署 DPI 的能力。对于给定的数据流集合 F 和节点集合 V，需要确定在哪些网络节点部署 DPI 引擎，既能满足集合 F 中所有数据流都经过 DPI 引擎进行检测，又能满足网络中传输所有数据流的总代价最小。每个节点的 DPI 能力受 CPU 处理能力的限制。每个节点的 DPI 处理能力用 C_d 表示，部署 DPI 功能的代价用 W_d 表示，网络链路带宽的代价用 W_{bw} 表示。

部署总代价是指使用网络资源代价与部署 DPI 引擎代价之和，设 $d_i^f=1$ 表示选择节点 i 对流 f 进行深度包检测，$d_i^f=0$ 表示不进行检测，数据流 f 的大小用 f_s 表示，其源和目的分别来源于给定数据流 f 在网络连接图中的起始节点和终止节点。链路分配状态用 $x_{i,j}^f$ 表示，$x_{i,j}^f=1$ 表示数据流 f 经过网络链路 (i,j)，$x_{i,j}^f=0$ 则表示不经过链路 (i,j)。用 $d_i=1$ 表示节点 i 具有 DPI 功能，$d_i=0$ 表示节点 i 不具有 DPI 功能。

代价最小化 DPI 引擎部署的线性规划形式如下：

最小化：

$$\mathrm{Min}\left(\sum_{(i,j)\in E,f\in F} f_s\times x_{i,j}^f\times w_{bw}+\sum_{i\in V} d_i\times w_d\right)$$

满足以下约束条件：

$$\sum_{f\in F} f_s\times x_{i,j}^f\leqslant C_{i,j},\ \forall\,(i,j)\in E\text{（链路能力约束）}$$

$$\sum_{i\in V} d_i^f=1,\ \forall\, f\in F\text{（深度包检测约束）}$$

$$\sum_{i\in V} f_s\times d_i^f=C_d,\ \forall\, f\in F\text{（DPI 引擎能力约束）}$$

$$\sum_{f\in F} d_i^f\leqslant d_i,\ \forall\, i\in V,\ \forall\, f\in F\text{（DPI 部署数量约束）}$$

该问题是一个完全 NP 问题，由于实现线性规划的复杂度较高，本文只在针对较少的节点时使用，以便查找最优的解决方案。

4　代价最小的 DPI 部署算法

本节提出一个代价最小的 DPI 部署算法（Cost Minimization Greedy algorithm，CMG），其主要目标是使部署 DPI 引擎的总代价最小。

用图 $G_n(V,E_n)$ 表示网络结构图，用 $F_t(V,E_t)$ 表示数据

流传输矩阵，V 表示网络节点的集合，E_n 表示网络中链路的集合，E_t 表示数据流经过的链路集合。网络节点 i 的承载带宽等于图 F_t 中所有数据流通过节点 i 的带宽大小之和，承载带宽数值越大，表示承载的传输数据量越大。承载带宽用 B_{ij} 表示，其中 i,j 为网络节点。

数据流传输矩阵 F_t 根据网络中数据流的统计信息生成，流统计信息包含源地址、目的地址和带宽，但在 F_t 中记录的源地址为数据流在网络图 G_n 中的第一个传输节点，目的地址则为数据流在网络图 G_n 中的最后一个传输节点。

定义 DPI 部署的总代价按式(1)进行计算：

$$cost=W_d\times di+W_{bw}\times(\sum_{(i,j)}B_{i,j}) \tag{1}$$

其中，$cost$ 表示代价值，DPI 表示 DPI 引擎的集合，W_d 表示部署 DPI 引擎的代价，di 表示部署 DPI 引擎的数量，W_{bw} 表示单位带宽资源的代价，$B_{i,j}$ 表示链路(i,j)的使用带宽。计算值包含两部分，第一部分为部署 DPI 的代价，第二部分为使用网络带宽资源产生的代价。第二部分为在网络中部署 DPI 引擎节点，并使得所有数据流至少经过一个 DPI 引擎节点，整个网络中所有节点承载数据流占用带宽的代价和。

两部分代价之和最小的目标函数，与线性规划中的目标函数一致，与网络带宽资源代价和部署 DPI 代价相关。提出基于代价最小化的启发式贪婪算法，在每一步部署一个新 DPI 引擎时，选取部署总代价最小的节点。代价最小贪婪算法(CMG)的具体描述如图 4 所示。

```
Cost Minimization Greedy algorithm
输入:网络拓扑 G(V,E),网络节点数 n,数据流集合 F
输出:DPI 部署节点集合 di
1. 初始化:部署节点集合 di={Φ},部署节点数 k=0,网络节点数
   n,部署代价 cost[k=0…n] =∞;
2. while k<n do{
3.    遍历所有未部署 DPI 节点 do{
4.      di =di+i;
5.      计算 F 经过 di 中节点的最短路径;
6.      计算 V 中所有网络节点传输的数据流带宽和;
7.      按式(1)计算部署节点 i 的代价值 costki;
8.      if costki<cost[k] then{
9.           cost[k]=costki,s=i;
10.     }end if
11.  }end 遍历所有未部署 DPI 节点;
12.  if cost[k]>cost[k-1] then{
13.     return di;
14.  }end if
15.  di=di+s;
16. }end while
17. return di
```

图 4 代价最小贪婪算法的伪代码

由于通过迭代方式选取部署 DPI 节点，当数据流传输规模较大且低于全网传输能力时，贪婪算法是近似公平的。对每一条流，该算法逐个考虑不同的 DPI 引擎部署，在计算源和目的之间的最短路径时，考虑到 DPI 引擎的约束，使用带约束最短路径优先算法。

在增加 DPI 引擎节点的过程中，如果 DPI 部署的总代价计算值高于上一次 DPI 部署的计算值，说明增加 DPI 引擎部署的代价高于节省的网络带宽资源代价，则代价最小贪婪算法终止，并返回上一次 DPI 部署；否则选取代价最小节点进行部署，从新的拓扑开始迭代计算。

5 优化最小代价部署算法

本节对上一节提出的代价最小贪婪算法进行优化，提出一个优化的代价最小贪婪算法(Optimized Cost Minimization Greedy Algorithm，OCMG)，其可减少每次遍历的计算量，但计算的 DPI 部署变化不大。定义优化代价最小贪婪算法 OCMG 的部署总代价如下：

$$cost=W_d\times di+W_{bw}\times(\sum_{(i,j)}B_{i,j}-\sum_{(i,j)}B^0_{i,j})$$

其中，第一部分为部署 DPI 引擎的总代价，与式(1)相同；第二部分为使用网络带宽资源代价的增加值。$\sum_{(i,j)}B_{i,j}$ 为部署 DPI 引擎节点后，所有数据流在网络中按照受 DPI 引擎节点约束的最短路径进行传输时的网络带宽资源；$\sum_{(i,j)}B^0_{i,j}$ 为所有数据流按照最短路径传输时的网络带宽资源，其实际上也是理想情况下的最小网络带宽资源代价，称为最小承载带宽。优化代价的最小贪婪算法的具体步骤描述如下：

(1)OCMG 在部署 DPI 节点前，首先计算所有数据流为最短路径时的最小承载带宽代价，在计算过程中记录每一个节点的承载带宽。

(2)在所有节点中选取承载带宽最大的节点作为部署 DPI 节点，这是由于该节点为网络中的关键节点。计算所有数据流经过该节点的最短路径，并记录各节点承载带宽与最小承载带宽条件下的差值。差值越大，说明 DPI 引擎对最短路径的影响越大，该节点越容易被选为下一个部署 DPI 节点。

(3)选取上一步骤中承载带宽与最小承载带宽条件下差值最大的节点，部署 DPI 引擎，并重新计算所有数据流经过该 DPI 引擎的最短路径，记录各节点承载带宽与最小承载带宽条件下的差值。重复进行，直至满足停止条件。

优化代价最小贪婪算法有两个终止条件：1)新增部署 DPI 节点时总代价反而增加，这种情况表明增加部署 DPI 节点的费用超过节省的网络带宽资源费用；2)当所有节点承载带宽与初始全部最短路径的承载带宽一致时，这种情况表明所有数据流均在最短路径上且具备 DPI 业务处理能力。

6 实验与结果分析

为了评估所提部署算法的性能，通过不同的网络实例对代价最小贪婪算法和线性规划算法进行了实验，本节给出模拟实验的过程和实验结果。

6.1 实验方法

使用 C 语言实现本文提出的线性规划算法(ILP)、代价最小贪婪算法(CMG)和优化代价最小贪婪算法(OCMG)。通过随机拓扑生成随机图模型[2]，对较小规模网络和大规模网络分别进行实验，较小规模网络主要用于验证 DPI 部署代价的影响，较大规模网络主要用于验证网络节点数的影响。

模拟实验使用的计算机的配置为 Intel 3240M CPU，主频 3.40GHz，4.0GB RAM，通过单线程实现。

6.2 小规模网络实验

随机生成图数据集，包含 20 个节点和 36 条 40G 链路，参照网络中应用生成 482 条数据流。图 5－图 7 为改变节点部署代价 W_d 的实验结果。从图 5 和图 6 可以看出，优化代价最小贪婪算法比线性规划运行速度快 9～20 倍，W_d 代价越大，代价最小贪婪算法的执行速度越优于线性规划。实际上，随着 W_d 的增加，线性规划算法的执行时间增加，但代价最小贪婪算法的执行时间反而减少，原因在于当节点部署价格 W_d 增加时，部署的 DPI 引擎数量减少，启发式算法的迭代次数更少。例如，一个节点部署价格为 10000 时，需要部署 4 个虚拟 DPI 节点，启发式算法只需要进行 5 次迭代，而线性规划的复杂度仍然不变。

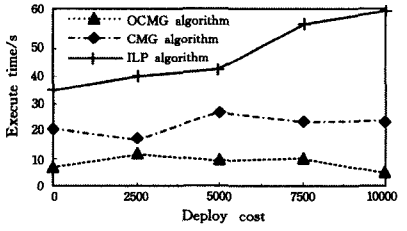

图 5　小规模网络下各算法执行时间的比较

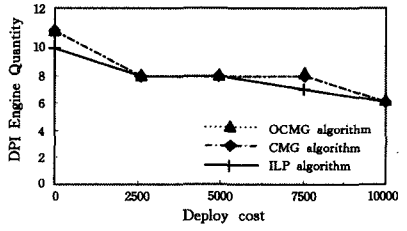

图 6　小规模网络下各算法 DPI 引擎数量的比较

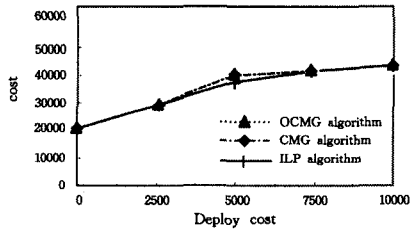

图 7　小规模网络下各算法总代价的比较

图 7 说明了 DPI 部署总代价的情况，总代价分为网络带宽代价和 DPI 部署代价两部分。从图 7 可以看出，代价最小贪婪算法的总代价接近于最优总代价，误差在 10%以内，两种算法都是对 DPI 部署代价和网络带宽代价的折衷。对于较小的 DPI 部署代价 W_d，总代价主要由部署 DPI 代价组成，当 W_d 增加时，使用带宽代价会占总代价的大部分，在这种情况下，越来越多的数据流从最短路径重路由到包含部署 DPI 节点的路径上，因此总代价增长并不剧烈。

6.3 大规模随机图实验

为了研究大规模实验网络，生成平均分布的随机网络图。由于网络中并没有广泛接受的模型，使用这种分布只是作为示例来生成不同结构的网络图。

生成的随机网络图中，随机增加任意两个节点之间的连接边，使整个网络图成为一个全连接图。生成的系列图中节点数递增，图中边的百分比不变，每一对节点之间的双向传输带宽为 10Gbps，数据流带宽按照 10Mbps 选取，虽然与实际并不相符，但目的是取平均情况，因此并不影响。DPI 部署代价为 1000。

图 8－图 10 示出了节点数量变化时代价最小贪婪算法和线性规划算法的实验对比结果。图 8 为执行时间对比结果，图 9 为部署 DPI 引擎的数量对比结果，图 10 为总代价对比结果。从图中可以看出，总代价和部署 DPI 节点数量具有较好的符合度，这是由于网络分布图中部分数据流最短路径存在重合节点，使得 DPI 业务的部署更容易受到约束。

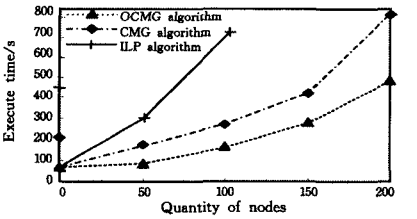

图 8　大规模网络下各算法执行时间的比较

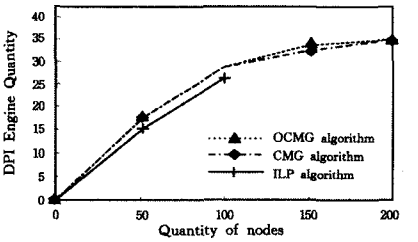

图 9　大规模网络下各算法 DPI 引擎数量的比较

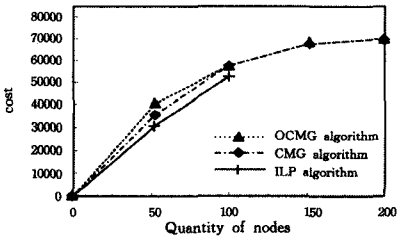

图 10　大规模网络下各算法总代价的比较

从图 8 可以看出，线性规划算法的执行时间远超过代价最小贪婪算法（CMG）和优化代价最小贪婪算法（OCMG）的执行时间。由于代价最小贪婪算法（CMG）和优化代价最小贪婪算法（OCMG）使用多次迪杰斯特拉（dijkstra）算法，对于 m 条边和 n 个节点的网络图，迪杰斯特拉算法的复杂度为 $O(m+n\log n)$，因此增加边和节点的数量会相应地增加执行时间。

以上实验结果表明，网络规模和 DPI 部署代价严重影响时间性能，当节点数量有限时，线性规划执行时间呈指数增长，这是两个代价综合作用的结果。

国内外学者对类似 NP 问题的求解，大多采用遗传算法、蚁群算法、粒子群算法等启发式算法，算法复杂度为 $O(n^2)$。一般情况下，单个域中 DPI 部署节点数量远小于网络节点数量，本文将多目标算法线性组合为单个目标，通过贪婪算法进

行计算，计算复杂度与迪杰斯特拉(dijkstra)算法相当，与启发式算法相比其性能有所改善。通过实验结果表明，优化代价最小贪婪算法(OCMG)无论对于较小规模网络，还是较大网络规模，都是近似最优的。

结束语 网络功能虚拟化和软件定义网络等新技术的出现，为我们提供了一系列传输监控和安全管理的工具。DPI 引擎可以进行虚拟化，并作为一个软件功能部署到通用设备上。虚拟化 DPI 在网络中的部署需要确定合适的部署位置，但在部署过程中需要考虑软件许可费用、节能和网络带宽等开销，存在部署代价约束。本文根据这些不同的约束条件，将虚拟化 DPI 部署问题形式化为代价最小问题，按照线性规划问题进行形式化，提出了一个基于代价最小的贪婪算法。最后，在算法有效性和可计算性方面，将其与线性规划进行了比较。提出的算法在 DPI 部署代价和网络链路带宽开销之间进行折衷，实验结果表明该算法可有效实现网络中 DPI 引擎的部署。

网络功能虚拟化使业务部署具有极大的灵活性。下一步的工作主要包括在测试床或者真实的网络环境中进行验证，充分考虑实际链路中带宽约束、DPI 处理能力的影响，进一步验证算法的有效性；同时，对近似算法的健壮性进行研究，在进行数据流重定向时，增加数据流时延的约束。

参 考 文 献

[1] AQAZI Z, TU C C, CHIANG L, et al. SIMPLE-fying middlebox policy enforcement using SDN[J]. ACM Sigcomm Computer Communication Review, 2013, 43(4): 27-38.

[2] BREMLER-BARR A, HARCHOL Y, HAY D, et al. Deep Packet Inspection as a Service[C]// The 10th International Conference on Emerging Networking Experiments and Technologies. Sydney, Australia, 2014: 271-282.

[3] COTRONEO D, DE SIMONE L, IANNILLO A K, et al. Network Function Virtualization: Challenges and Directions for Reliability Assurance[C]// IEEE International Symposium on Software Reliability Engineering Workshops, 2014. Naples, Italy, 2014: 37-42.

[4] ETSI. Network functions virtualization introductory white paper [OL]. http://portal.etsi.org/NFV/ NFV_White_Paper.pdf.

[5] JIANG W J, LAN T, HA S, et al. Joint VM placement and routing for data center traffic engineering[J]. Infocom Proceedings IEEE, 2012, 131(5): 2876-2880.

[6] RAJAGOPALAN S, DAN W, JAMJOOM H, et al. Split/merge: System support for elastic execution in virtual middleboxes[C]// Usenix Conference on Networked Systems Design & Implementation, 2013. Lombard, Italy, 2013: 227-240.

[7] GEMBER A, PRABHU P, GHADIYALI Z, et al. Toward software-defined middlebox networking[C]// 11th ACM Workshop on Hot Topics in Networks, 2012. New York, USA, 2012: 7-12.

[8] SHERRY J, HASAN S, SCOTT C, et al. Making middleboxes someone else's problem: network processing as a cloud service [J]. ACM Sigcomm Computer Communication Review, 2012, 42(4): 13-24.

[9] LU G H, MIAO R, XIONG Y Q, et al. Using CPU as a traffic coprocessing unit in commodity switches[C]// First Workshop on Hot Topics in Software Defined Networks, 2012. Levin, New Zealand, 2012: 31-36.

[10] GRINGOLI F, ESTE A, SALGARELLI L. MTCLASS: Traffic classification on high-speed links with commodity hardware[C]// IEEE International Conference on Communications, 2012. Ottawa, Canada, 2012: 1177-1182.

[11] MEHRAGHDAM S, KELLER M, KARL H. Specifying and placing chains of virtual network functions[C]// IEEE 3rd International Conference on Cloud Networking, 2014. Luxembourg, 2014: 7-13.

[12] BOUET M, LEGUAY J, CONAN V. Cost-Based Placement of Virtualized Deep Packet Inspection Functions in SDN[C]// IEEE Military Communications Conference, 2013. San Diego, Canada, 2013: 992-997.

[13] CHAUDET C, FLEURY E, RIVANO H, et al. Optimal positioning of active and passive monitoring devices[J]. IEEE Review, 2005, 51(10): 71-82.

(上接第 85 页)

[7] YU C H, DOPPLER K, RIBERIRO C B, et al. Resource sharing optimization for device-to-device communication underlaying cellular networks[J]. IEEE Transactions on Wireless Communications, 2011, 10(8): 2752-2763.

[8] TAMURA H, SENGOKU M, NAKANO K, et al. Graph theoretic or computational geometric research of cellular mobile communications[C]// IEEE International Symposium on Circuits and Systems. Orlando, 1999: 153-156.

[9] ZHANG R, CHENG X, YANG L, et al. Interference-Aware graph based resource sharing for device-to-device communications underlaying cellular networks[C]// IEEE Wireless Communications and Networking Conference. Shanghai, China, 2013: 140-145.

[10] ZHANG H, SONG L, HAN Z. Radio resource allocation for device-to-device underlay communication using hypergraph theory [J]. IEEE Transactions on Wireless Communications, 2016, 15(7): 4852-4861.

[11] ZHANG H, WANG T, SONG L, et al. Graph-based resource allocation for D2D communications underlaying cellular networks [C]// IEEE/CIC International Conference on Communications in China-Workshops. Xi'an, China, 2013: 187-192.

[12] QIAN C, QIAN L P, WU H, et al. System throughput optimization for hybrid device-to-device cellular networks[J]. Computer Science, 2016, 43(1): 145-148, 177. (in Chinese)
钱程，钱丽萍，武航，等. 混合 D2D 蜂窝网络的系统吞吐量优化[J]. 计算机科学，2016，43(1)：145-148，177.

[13] HOANG T D, LE L B, LE-NGOC T. Resource allocation for D2D communication underlaid cellular networks using graph-based approach[J]. IEEE Transactions on Wireless Communications, 2016, 15(10): 7099-7113.

[14] ITU-R. Guidelines for Evaluation of Radio Interface Technologies for IMT-Advanced: ITU-R M. 2135[R]. 2008.