Software Engineering and Applications 软件工程与应用, 2016, 5(2), 122-129
Published Online April 2016 in Hans. http://www.hanspub.org/journal/sea
http://dx.doi.org/10.12677/sea.2016.52014

Hans汉斯

Research on the Mechanism of Virtual Router Control Plane Traffic Isolation Based on Historical Cost

Ke Chen, Xiaozhe Zhang, Xianming Gao

School of Computer Science and Technology, National Defense Science and Technology University, Changsha Hunan
Email: 285847745@qq.com

Received: Mar. 23rd, 2016; accepted: Apr. 6th, 2016; published: Apr. 12th, 2016

Abstract

In order to ensure the fairness of the virtual router to control the bandwidth resource of the planar link and the traffic isolation, in this paper, we design and implement the middle tier of communication, VNETDEV kernel module, between the virtual control plane and the logical forwarding engine based on the Linux system. On this basis, this paper proposes a traffic isolation mechanism in virtual control plane instances based on the historical cost. Through arranging the VNETDEV kernel module under the LinuxLXC virtual machine environment, verification of flow test for multiple virtual control plane instances can ensure the fairness and isolation of traffic between each virtual control plane.

Keywords

Virtual Router, Virtual Control Plane, Traffic Isolation, Historical Cost

基于历史开销的虚拟路由器控制平面流量隔离机制研究

陈　科，张晓哲，高先明

国防科学技术大学计算机学院，湖南 长沙

文章引用: 陈科，张晓哲，高先明. 基于历史开销的虚拟路由器控制平面流量隔离机制研究[J]. 软件工程与应用, 2016, 5(2): 122-129. http://dx.doi.org/10.12677/sea.2016.52014

Email: 285847745@qq.com

收稿日期：2016年3月23日；录用日期：2016年4月6日；发布日期：2016年4月12日

摘　要

为了保障虚拟路由器控制平面链路带宽资源的公平性以及流量隔离，本文在Linux系统上设计并实现了虚拟控制平面实例与逻辑转发引擎间的通信中间层——VNETDEV内核模块。在此基础上本文提出了一种基于历史开销的虚拟控制平面实例间流量隔离机制。通过在LinuxLXC虚拟机环境下部署VNETDEV内核模块，对多个的虚拟控制平面实例进行的流量测试验证能够保证各虚拟控制平面实例间的流量公平性和隔离性。

关键词

虚拟路由器，虚拟控制平面，流量隔离，历史开销

1. 引言

从上世纪九十年代开始，互特网经历了爆发式发展，已经成为社会人文科学和自然技术科学研究的基础，在各种方面的应用层出不穷，比如视频聊天、远程教育等[1]。互联网已经演变成为一种非常庞大的应用型网络，其商业价值尤为突出。比如 2015 年的双十一，淘宝天猫的点击量达到了 12 万次每秒，如果没有高效、安全、移动性能好的网络，如此庞大的短时的数据交互是不可能实现的。随着网络应用的增多，应用面的不断扩大，对网络的要求也不断的提高。在解决网络的移动性、安全性和服务质量等诸多问题时，往往不会采用设计一种更加高能的新的架构的网络体系，而是采用在原来网络的基础上进行修补的方式来解决。然而这样的做法并不能有效解决越来越多的网络需求，反而阻碍了创新性网络的研究与部署。

为了破解当前互联网在物理资源利用率低等方面的问题，在系统虚拟化技术思想的基础上[2]，以往的研究人员提出了一种将虚拟化技术运用到传统网络架构的思想，并逐渐形成和完善了网络虚拟化技术[3] [4]。在网络虚拟化中，虚拟路由器作为重要支撑点，使若干虚拟路由器实例运行在同一物理平台，实现它们的并行运行、相互隔离。虚拟路由器通常包括虚拟控制平面和虚拟转发平面两部分。其中，虚拟控制平面采用系统虚拟化技术，使得每个虚拟机中运行一个逻辑控制平面实例；虚拟转发平面承载多个逻辑转发引擎，每个逻辑转发引擎负责虚拟路由器内的分组报文转发任务，逻辑转发引擎与其对应的控制平面实例间通过共享的物理控制通路来实现协议报文和控制信息的传递。

虚拟路由器控制平面通常采用虚拟机技术来保障各个虚拟控制平面实例间的 CPU、主存和存储资源的隔离性。通过配置虚拟机的参数来分配每个虚拟控制平面实例占用的 CPU 核数、内存大小和文件存储空间大小，并保证上述资源间的严格隔离。但是，在多个虚拟控制平面实例共享同一物理板卡时，与虚拟转发平面逻辑转发引擎间的通信需共享同一物理控制通路。当某些虚拟控制平面实例占用物理接口带宽过大时，会影响其它虚拟控制平面实例的通信带宽，极端情况下会造成其它虚拟控制平面实例的控制通路丢包，甚至会造成虚拟控制实例与逻辑转发引擎间通信中断。目前，已有的 LXC [5]、KVM [6]、XEN [7]等虚拟机技术都没有保障虚拟控制平面实例间控制流量隔离的有效手段。

据此，本文在 Linux 系统上设计并实现了虚拟控制平面实例与逻辑转发引擎间的通信中间层——

VNETDEV 内核模块，在此基础上提出了一种基于历史开销的虚拟控制平面实例间流量隔离机制。通过为每个控制平面实例分配单独的通信队列，基于每个控制平面实例最近占用物理接口的开销来动态计算各队列中报文调度顺序，保证了虚拟控制平面实例间占用接口带宽资源公平性，实现了虚拟控制平面实例间的流量隔离。通过在 LinuxLXC [8] [9]虚拟机环境下部署 VNETDEV 内核模块，对多个的虚拟控制平面实例的流量测试验证，能够保证各虚拟控制平面实例间的流量公平性和隔离性。

2. 相关研究

针对虚拟路由器资源调度的相关研究，研究者提出了多种解决方案。在研究虚拟路由器的资源调度算法中，系统虚拟化工具起到了很重要的作用。主要的系统虚拟化工具 VMware ESXi，Xenserver，Hyper-V 等。以典型的系统虚拟化工具 Xen 为例，其主要的资源调度算法有三种，BVT (Borrowed Virtual Time) 调度算法[10]、SEDF (Simple Earliest Deadline First)调度算法[11]、以及 Credit 调度算法[12]。

以上算法都没有很好的解决虚拟路由器资源共享的公平性，也没用对数据量的隔离性进行考虑。为了解决虚拟路由器流量公平性与隔离性，就必须改进现有的资源调度算法或提出新的资源调度算法来满足虚拟路由器的设计需求。我们在前期工作基础上[13]，将基于转发开销的资源调度算法扩展到解决控制链路公平性的问题。

3. 虚拟控制平面框架

虚拟控制平面主要功能是用于承载若干控制平面实例，并且保证这些控制平面实例的独立性和隔离性。目前，虚拟控制平面通常采用系统虚拟化工具，将控制平面实例部署在对应的虚拟机中。通过配置虚拟机的参数来确保控制平面的独立性，确保 CPU 和内存资源的独占性。与此同时，控制平面实例通过网络接口与对应的逻辑转发引擎进行通信，这些控制平面实例共享链路带宽。然而，现有的机制却无法保证控制平面实例共享链路带宽的独立性。为此，在 Linux 系统上设计并实现了虚拟控制平面实例与逻辑转发引擎间的通信中间层——VNETDEV 内核模块，如图 1 所示。

从图中可以看出，控制平面实例与外界(即逻辑转发引擎)间通信都需要经过 VNETDEV 模块。VNETDEV 模块主要功能包括：一、它支持虚拟接口的创建与删除功能，并将指定的虚拟接口分配给控制平面实例；二、它是虚拟控制平面实现控制平面实例流隔离的重要组成部分。

为了确保链路带宽资源的公平性，VNETDEV 模块中为每个控制平面实例分配一个队列集，包括一个入队列和一个出队列。入队列中用于存放网络接口接收到的报文；而出队列中用于存放控制平面实例下发的报文。而报文调度器则决定了入队列中报文的调度次序以及从控制平面实例接收报文的时间。它通过报文调度的顺序来保证链路带宽资源的公平性。

4. 基于历史开销的报文调度算法

为了保障链路带宽资源的公平性，我们在报文调度器中设计基于历史开销的报文调度(Historical Cost Based Packet Scheduling Algorithm, HC-PSA)算法。它通过统计报文最近几次的调度开销来确定队列中报文调度的优先级，从而确保若干控制平面间流量隔离。HC-PSA 算法能依据 VNETDEV 模块的报文调度开销历史数据计算出队列中报文调度优先级，然后根据报文的优先级向控制平面实例推送数据包，从而确保链路带宽资源共享的公平性。

4.1. 优先级的计算

在设计 HC-PSA 算法时，第一步是确定影响队列中报文调度优先级的因素。在确定因素的过程中，如果仅考虑最主要因素，可能会使最终计算结果存在偏差，而无法正确反映出控制平面实例对链路带宽

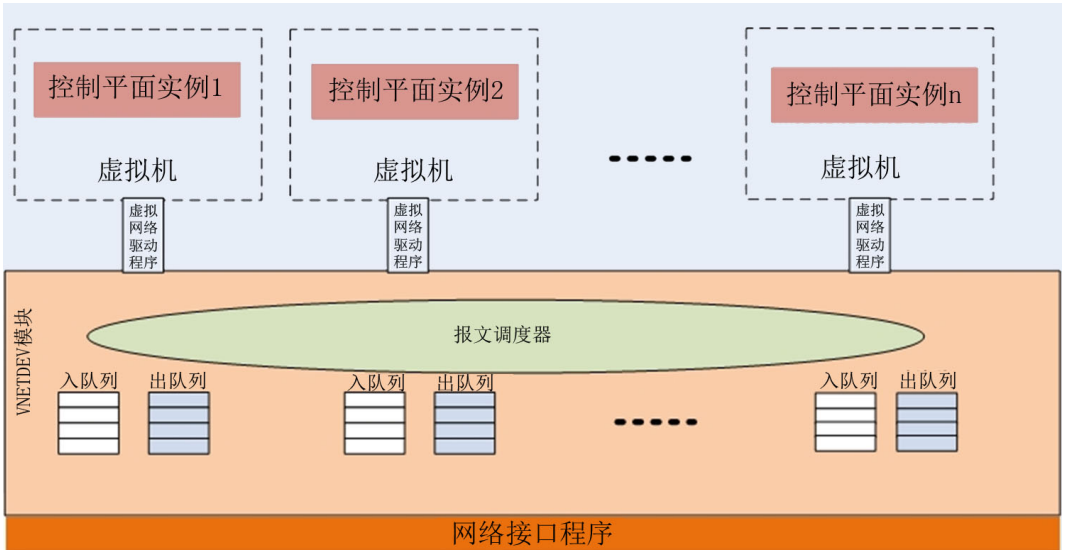

Figure 1. Virtual control plane frame
图 1. 虚拟控制平面框架

资源使用的实际情况。但某些对计算结果影响十分微小的因素如果考虑在内，又会增加计算的复杂性，而且不能清晰的反映出优先级计算的主要因素。

结合控制平面实例的特性，HC-PSA 算法考虑的因素有：1) 控制平面实例占用链路带宽资源的权值；2) 入队列中报文等待处理的时间；3) VNETDEV 模块中控制平面实例最近 a 次的报文调度开销；4) 控制平面实例的延迟敏感度；5) 报文是否由于处理时间等问题是否被丢弃。那么，HC-PSA 算法中队列中报文调度的优先级的计算公式，如公式(1)所示。

$$Q_i = -\log\left(\gamma_i * \frac{A_i[n]}{T_i} * \frac{1}{\overline{A_i[a]}}\right)\left(A_i[n] + f_i\left(\overline{A_i[a]}, L\right) < T_i\right) \tag{1}$$

其中，Q_i 表示第 i 个的入队列中报文调度优先级；γ_i 表示第 i 控制平面实例的带宽权值；$A_i[n]$ 表示第 i 控制平面实例对应的报文等待处理的时间；T_i 表示第 i 个控制平面实例容忍的最大处理时延；$\overline{A_i[a]}$ 表示第 i 个控制平面实例最近 a 条报文调度开销的均值；$f_i\left(\overline{A_i[a]}, L\right)$ 表示第 i 个控制平面实例下一报文预测处理时间；L 表示报文的长度。

4.2. 历史开销

$\overline{A_i[a]}$ 表示控制平面实例在最近一段时间的报文调度开销的均值。它能够使平均吞吐量较低的控制平面实例增大被调度的机率。报文调度开销的均值对优先级有明显的影响，能够反映出队列中报文调度的优先级，计算公式如公式(2)所示：

$$\overline{A_i[a]} = \frac{\sum_{i=1}^{a} t_i[n]}{a} \tag{2}$$

其中，$t_i[n]$ 表示第 i 个控制平面实例最近第 n 次调度开销。a 值表示控制平面实例最近调度报文的次数。由式(2)可以看出 a 值与报文调度开销的均值成反相关。当最近一段时间数据流波动比较大时，如果 a 过大，$\overline{A_i[a]}$ 则不能有效反映控制平面实例的调度报文开销；当控制平面实例在使用前处于空闲状态时，如果 a 过小，也会致使 $\overline{A_i[a]}$ 报文转发开销的均值失真。这两种情况都不能有效地保证链路带宽资源共享的

公平性。因此，为了更好的体现出 HC-PSA 算法的优越性，必须选择一个比较合理的阈值 a。

4.3. 前提条件

如果报文在还未处理前已经被丢弃，那么计算优先级就不存在任何的意义，所以在报文还未处理前决定报文是否被丢弃是优先级计算的前提条件。需要判断报文是否被丢弃，通过比较报文等待时间和报文处理时间之和与控制平面实例预设的处理时延即可得出。当报文等待时间和报文处理时间之和小于控制平面实例预设的处理时延时，则计算控制平面实例的优先级；否则将该报文丢弃，避免造成系统不必要的开销。前提条件用公式可表示为：

$$A_i[n]+f_i\left(\overline{A_i[a]},L\right)<T_i \tag{3}$$

在式(3)中，控制平面实例容忍的最大处理时延 T_i 与报文的等待处理时间 $A_i[n]$ 是已知的。那么，$f_i\left(\overline{A_i[a]},L\right)$ 便是处理报文是否被丢弃的唯一的一个变量参数。其计算公式为：

$$f_i\left(\overline{A_i[n]},L\right)=k*L*\frac{\overline{A_i[a]}}{\overline{L_i[a]}} \tag{4}$$

其中，k 为与物理平台有关的系数；$\overline{L_i[a]}$ 表示最近 a 个报文的平均长度；L 表示当前待处理的报文长度；$\overline{A_i[a]}$ 表示最近 a 条报文调度开销的均值。

4.4 偏差量计算

在 HC-PSA 算法中，偏差量是指虚拟路由器中控制平面实例实际占有的共享资源与理论上可以占用的共享资源的均方差。现假设偏差量为 θ，控制平面实例 i 实际的调度报文开销为 T_r，控制平面实例理论上的调度报文开销为 T_t。那么，偏差量的计算公式为：

$$\theta=\sum_{i=1}^{N}\left(\frac{T_r(i)-T_t(i)}{T_t(i)}\right)^2 \tag{5}$$

偏差量 θ 的大小与实际的调度报文开销与理论上的调度报文开销的差值成正相关。实际值越接近理论值说明偏差量越小，链路带宽也就越公平；而实际值与理论值相差越大说明偏差量就会越大，链路带宽的公平性也就越差。由公式(2)与公式(5)可以看出，通过调节阀值 a，可以有效的调节偏差量 θ。反过来，当偏差量 θ 越大时，a 不能真实体现链路带宽使用的真实情况，需要调整 a 值的大小。

HC-PSA 算法主要特点：第一，通过统计 VNETDEV 模块中最近 a 次报文调度开销，计算出队列中报文调度的优先级，将高优先级的报文发送到指定的控制平面实例。第二，HC-PSA 算法能够预测报文处理时间，通过与控制平面实例中预设的处理延时作对比，预先决定报文的处理或丢弃，这样可以有效降低处理报文时的额外开销。第三，通过计算优先级的公式中的延迟值，以达到控制平面实例对延时敏感强度的要求。

5. HC-PSA 算法的描述

HC-PSA 算法中报文调度开销的均值反映了当前控制平面实例的调度报文的平均开销，并尽可能保证每一个控制平面实例所占用的链路带宽资源是公平的。它根据控制平面实例历史数据流来调整队列中报文调度的优先级，确保了低吞吐量的逻辑控制引擎获得报文的机会，从而保证了链路带宽的公平性。HC-PSA 算法的具体描述如图 2 所示。

HC-PSA 算法描述
输入：入队列存在报文
输出：推送数据包到对应的控制平面实例
过程：
while(遍历每个入队列中头报文) {
Next：计算出$A_i[n]+f_i(\overline{A_i[a]}, L)$;
if $(A_i[n]+f_i(\overline{A_i[a]}, L) < T_i)$
计算出$\overline{A_i[a]}$;
计算Q_i;
else
丢弃该报文，读取入队列中下一报文;
If(!null)
goto Next;
}
max $\{Q_0、Q_1 \cdots Q_i\}$;
将优先级最高的报文推送到对应控制平面实例;

Figure 2. Specific description of the algorithm
图 2. 算法的具体描述

通过 HC-PSA 算法描述可知：当入队列存在报文时才能触发 HC-PSA 算法。它首先利用前提条件预测待处理报文是否被丢弃，如果预判报文被丢弃，则读取入队列中下一条报文；否则计算出控制平面实例的报文调度开销的均值和调度优先级。最后，通过比较得出优先级最高的队列中报文，并将报文推送到对应控制平面实例中。HC-PSA 算法与其他调度算法最大差别在于前者通过计算控制平面实例的报文调度开销的均值来更加精确地确定队列中报文调度次序，从而保证了链路带宽资源共享的公平性。

6. 实验结果与分析

6.1. 实验环境

虚拟控制平面采用的系统虚拟化工具为 LXC，构建不同的容器，并将对应的控制平面实例部署在对应的容器中。VNETDEV 模块采用多线程、多队列的编程方式。虚拟控制平面运行在一台 IBM 服务器上为了验证基于历史开销的控制平面实例的流量隔离机制的有效性，在原型平台上创建了 3 个控制平面实例 A、B、C，它们占用的链路带宽额定值为 50 Mbps、40 Mbps 和 30 Mbps。在实验过程中，分别向三个控制平面实例发送测试数据流如图 3 所示。

在 0 时刻引入的数据流均略高于它们的链路带宽额定值。在第 5 s~10 s，控制平面实例 A 的数据流保持不变，控制平面实例 B 的数据流下降到其链路带宽额定值 40 Mbps，控制平面实例 C 的数据流下降到 20 Mbps。在 10 s 以后，控制平面实例 A 和 C 保持不变，控制平面实例 B 的数据流略有提高。

根据上文所述算法，在以上所述数据流的变化情况下，控制平面实例 A、B、C 所对应的数据流占用带宽应该更趋于公平合理，经过数据采集，控制平面实例 A、B、C 占用带宽的实际量确实更加的公平合理。

6.2. 链路带宽的公平性

如图 4 所示，为控制平面实例 A、B、C 实际数据流量图。在 0 s~5 s，控制平面实例 A、B、C 的实

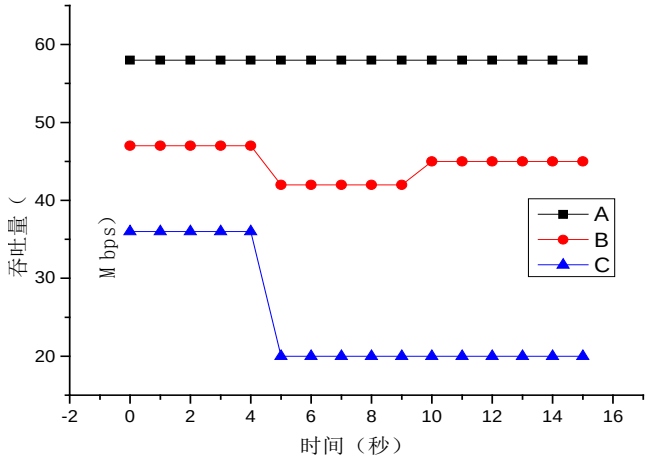

Figure 3. Test flow
图 3. 测试流

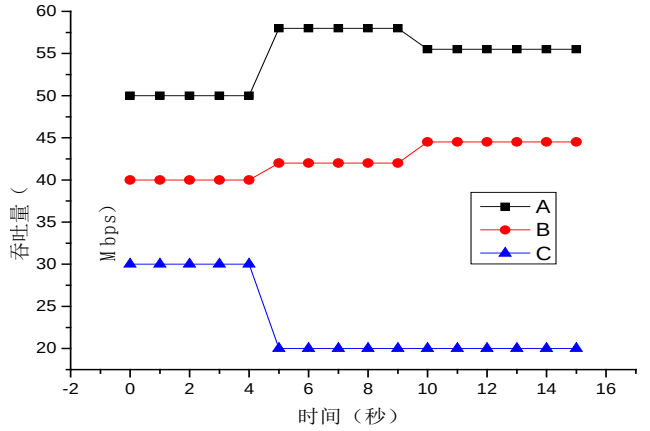

Figure 4. Actual data quantity
图 4. 实际数据量

际数据量为其链路带宽的额定值。在 5 s~10 s，控制平面实例 A 的实际数据量明显高于其数据链路带宽的额定值，控制平面实例 B 的实际数据量略高于其数据链路带宽的额定值，控制平面实例 C 的实际数据量则为其引入数据量的值。在 10 s 以后，控制平面实例 A 的实际数据量略有下降，但仍然高于其链路带宽的额定值；控制平面实例 B 的实际数据量比 5 s~10 s 内的数据量略有提高；控制平面实例 C 的实际数据量则仍然保持不变。

从以上数据可以得出，在控制平面实例 B、C 引入的数据量减小时，控制平面实例 A 的实际数据量有明显增加。在控制平面实例 B 引入的数据量有所提升时，控制平面实例 A 引入数据所占用的带宽有所下降，以供给控制平面实例 B 引入数据量的需求。实验结果反映出 HC-PSA 算法能够满足数据流占用带宽的公平性要求。在整个过程中，并没有因为控制平面实例 A、B 的数据流占用了更多的带宽而使得控制平面实例 C 引入的数据流发生异样的变化，没有发生丢包、接收或发送报文时延过大等情况，说明 HC-PSA 算法能够满足控制平面实例间流量隔离。

6.3. 偏差量与 *a* 值关系

如图 5 所示，*a* 值在 0~20 之间时，随着 *a* 值的增加，偏差量逐渐在减小。*a* 值在 20~50 之间时，偏差量趋于平稳。说明了调节 *a* 值可以达到调节偏差量的目的。

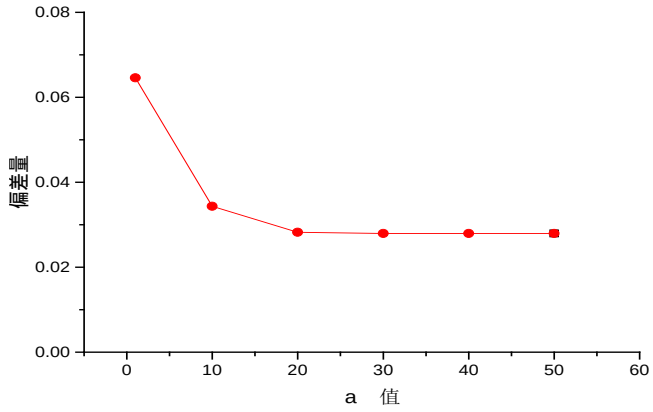

Figure 5. The relationship between deviation and *a* value
图 5. 偏差量与 *a* 值关系

7. 总结与展望

由于虚拟路由器支持多个控制平面实例并行运行在同一物理平台，这使得多个控制平面实例在接收数据时，会产生抢占接口带宽的问题。如何保证占用带宽的公平性是当前迫切需要解决的问题。本文针对现有的资源调度算法存在的不足，提出了一种基于历史开销的控制平面实例的流量隔离机制(HC-PSA)。它为每个控制平面实例分配单独的队列，基于每个控制平面实例最近占用物理接口的开销来确定调度队列中报文的顺序。通过队列中报文调度顺序来保证控制平面实例占用接口的带宽资源，从而实现控制平面实例间流量隔离。实验结果表明：基于历史开销的控制平面实例的流量隔离机制能严格地保证各个控制平面实例占用接口的带宽的独立性。

参考文献 (References)

[1] Anderson, T., Peterson, L., Shenker, S. and Turner, J. (2005) Overcoming the Internet Impasse through Virtualization. *Computer*, **38**, 34-41. http://dx.doi.org/10.1109/MC.2005.136

[2] 英特尔开源软件技术中心. 系统虚拟化[M]. 北京: 清华大学出版社, 2009.

[3] Mosharaf Kabir Chowdhury, N.M. and Rauof, B. (2010) A Survey of Network Virtualization. *Computer Network*, **54**, 862-876. http://dx.doi.org/10.1016/j.comnet.2009.10.017

[4] 高先明, 王宝生, 张晓哲, 马世聪. 面向网络虚拟化技术的管控[J]. 北京邮电大学学报, 2015, 38(6): 120-126.

[5] Helsley, M. (2009) LXC: Linux Container Tools. IBM developerWorks: Technical Library.

[6] Rosenblum, M. (1999) VMware's Virtual Platform™. *Proceedings of Hot Chips*, 185-196.

[7] Barham, P., Dragovic, B., Fraser, K., Hand, S., Harris, T., Ho, A., *et al*. (2003) Xen and the Art of Virtualization. *ACM SIGOPS Operating Systems Review*, **37**, 164-177. http://dx.doi.org/10.1145/1165389.945462

[8] Calarco, G. and Casoni, M. (2012) On the Effectiveness of Linux Containers for Network Virtualization. *Simulation Modelling Practice & Theory*, **31**, 169-185. http://dx.doi.org/10.1016/j.simpat.2012.11.007

[9] Zakić, A.S., Veinović, M., Trajković, S., *et al*. (2015) Linux Containers: Docker Platform for Distributed Applications. Infotech.

[10] Duda, K.J. and Cheriton, D.R. (1999) Borrowed-Virtual-Time (BVT) Scheduling: Supporting Latency-Sensitive Threads in a General-Purpose Scheduler. *ACM SIGOPS Operating Systems Review*, **33**, 261-276.

[11] Gu, Z. and Zhao, Q. (2012) A State-of-the-Art Survey on Real-Time Issues in Embedded Systems Virtualization. *Journal of Software Engineering & Applications*, **5**, 277-290. http://dx.doi.org/10.4236/jsea.2012.54033

[12] Soltesz, S., Tzl, H., Fiuczynski, M.E., Bavier, A. and Peterson, L. (2007) Container-Based Operating System Virtualization: A Scalable, High-Performance Alternative to Hypervisors. *ACM Sigops Operating Systems Review*, **41**, 275-287. http://dx.doi.org/10.1145/1272998.1273025

[13] 高先明, 张晓哲, 王宝生, 卢泽新, 马世聪. 面向虚拟路由器的基于历史转发开销的资源调度算法[J]. 电子与信息学报, 2015, 37(3): 686-692.