多核平台入侵检测系统负载均衡算法设计与实现*

李彦君，钟求喜，陈 诚，陆华彪
（国防科学技术大学 计算机学院，长沙 410073）

摘 要：负载均衡是基于多核平台实现高速入侵检测系统的关键技术之一。基于真实流量统计分析发现的流阈值与流数目、流字节数之间变化的规律，提出只调整较大流的动态分流算法 HCLF，并实现了原型系统。实验测试表明，与静态哈希算法和新流调整算法相比，HCLF 算法在负载均衡度、系统丢包率方面具有显著的优越性，改善了多核平台高速入侵检测系统对突发流量和应用环境的适应性。

关键词：多核平台；入侵检测系统；CPU 利用率；较大流；负载均衡度；丢包率
中图分类号：TP309 **文献标志码**：A **文章编号**：1001-3695(2012)04-1413-04
doi:10.3969/j.issn.1001-3695.2012.04.059

Design of load balancing algorithm for IDS based on multi-core platform

LI Yan-jun, ZHONG Qiu-xi, CHEN Cheng, LU Hua-biao
(*School of Computer Science, National University of Defense Technology, Changsha* 410073, *China*)

Abstract: Load balancing is one of the key technologies of designing high-speed intrusion detection system (IDS) based on multi-core platform. In terms of statistic analysis to the real Internet traffic, this paper found the law about the threshold of flows, the number of flows and the number of their bytes. It proposed a novel load balancing algorithm (named HCLF), which adjusted only the large flows. After the antitype system was implemented, the experiments show that HCLF has the distinct advantage on load balancing metric and packet loss rate, compare with the static hash algorithm and the adjusting new flow algorithm. It improves the adaptability of the high-speed IDS based on multi-core platform to sudden load and environment.
Key words: multi-core platform; IDS; CPU utilization; large flow; load balancing metric; packet loss rate

随着网络流量的快速增长，入侵检测系统的处理性能成为网络安全防护发展的瓶颈之一。当前，通用多核平台的计算能力不断提升，为设计实现高性价比入侵检测系统提供了有利契机。作为性能提升的关键技术，多核负载均衡算法在负载均衡度、动态自适应性、数据流破坏率、算法开销等方面面临着更高要求，成为高性能入侵检测研究的重点。

本文通过分析我国互联网骨干链路的真实流量，发现了流阈值与流数目、流字节数之间变化的规律，并选取一些数目少但所占的流量比重却很大的流定义为较大流。已有的负载均衡算法大多侧重于数据流数目的均衡，可能将多个较大流划分到同一处理核。当系统整体负载较高时，这会造成单个处理核心过载，进而导致入侵检测系统丢包乃至漏报。为此，本文提出了以处理核的 CPU 利用率为依据，当其超过一定阈值时，动态调整较大流的负载均衡算法 HCLF(Hashing considering CPU utilization rate and large flows)。

1 相关研究

入侵检测系统的处理能力一直受到网络流量快速增长的挑战，对于多处理器搭建的入侵检测系统，高效的负载均衡算法是应对这一挑战的关键，国内外众多专家学者为此做了大量的研究和探索[1~5]。

程光等人[1]通过实验验证了循环位移三位的平均随机性最大，提出了位移三位然后异或的 SHIFT_XOR_3 算法。刘许刚等人[2]从信息熵的角度分析异或移位(SHIFT-XOR)算法，并针对网络数据的处理，进一步优化了哈希算法的散列性能。他们研究的都是静态分流算法。静态分流算法实现简单，但缺乏实时控制突发流量的能力。

余颖等人[3]通过观察各个探测器上数据包的进出情况，由包预测负载均衡算法预测下一个时刻各探测器上的负载情况，避免了将新连接加入到流量突发探测器的可能，提高了负载均衡的效率。

Shi 等人[4]分析网络中 IP 地址的分布近似符合齐夫定律，提出了在处理引擎之间调节极大流(流量中流量最大的前几个流)的 SHI 算法，将属于极大流的报文散射到负载最轻的处理引擎中。但是极大流所占流量比重偏小，该算法动态调整力度不够。

于洪伟[5]提出了基于流的动态负载均衡(MDLB)算法，算法维护两张表，即检测引擎实时负载表和数据包下发表。但两个表的表项过多且没有经过优化，每个报文都导致对表元素完全遍历，维护两张表非常耗费资源；并且数据包下发表是根据新流的到来进行更新的，但是高速网络中单位时间内新流的个数是难以预测的，数据包下发表的长度该如何确定关系到整个系统的可行性和准确性，作者未给出明确说明。

总体上来说，现有的高速环境下入侵检测负载均衡算法偏

收稿日期：2011-10-07；**修回日期**：2011-12-09 **基金项目**：国家自然科学基金资助项目(61003303)

作者简介：李彦君(1982-)，男，山西武乡人，硕士研究生，主要研究方向为网络安全(liyanjun7441@sina.com)；钟求喜(1969-)，男，副研究员，博士，主要研究方向为网络与信息安全；陈诚(1985-)，男，硕士研究生，主要研究方向为网络安全；陆华彪(1983-)，男，博士研究生，主要研究方向为网络安全.

重于研究动态调整的策略、流量的具体分割方法等,从网络流量特性方面研究入侵检测分流的较少。本文从分析网络流量特性入手,根据统计分析得出的规律,提出应用于通用多核平台的动态分流算法 HCLF。

2 HCLF 算法

2.1 网络流量特性

为了保证开发的网络入侵监测系统可以应用于实际网络环境,笔者在我国互联网络某省级骨干链路通过光旁路复制采集到真实流量若干组,采集时间为 2010 年 10 月。

文献[6]中作者采用 NLAR PMA 组在 Internet2 上获取的网络流量若干组进行实验,发现流量中存在一些流数目少、但其所占的流量比重却很大的现象,称这些流为较大流,并指出较大流数目与比重变化稳定,对流量阈值变化不太敏感[6]。本文认为作者只是指出较大流现象,并没有完整地揭示出取不同的流阈值,流数目、流字节数有怎样的变化规律以及较大流该如何选取。本文决定以采集到的真实流量为对象作进一步统计分析,寻找流阈值与流数目、流字节数之间变化的规律,为下一步工作做准备。

本文中流是指在同一网络环境下、一段时间内满足某些特定条件的一定数量报文的集合。这些报文应该满足如下条件:一条流可用五元组标志:〈PN,SIP,SPT,DIP,DPT〉,分别代表协议号、源 IP 地址、源端口号、目的 IP 地址、目的端口号,需要注意这些报文 PN 相同,但是源 IP 地址、源端口号和目的 IP 地址、目的端口号这两组信息可以相同也可以对调,这是由于进行会话的双方源、目地址具有相对性。对于无连接协议(UDP、ICMP 等)的数据包以及不完整 TCP 连接的数据包,相近时间内具有相同或源、目对调后相同的五元组标志〈PN,SIP,SPT,DIP,DPT〉的所有报文组成为一条流。

本文设计的流分析器按周期(接收 100 000 个报文)统计如下数字:每个周期内流的数目、字节数大于设定阈值的流的数目以及这些流的字节总数。在采集到的流量数据包中,在不同的位置随机抽取了多个周期进行统计分析,结果如下(由于是统计规律,以下结论使用约数):

a)各周期内流的总数十分相近,数目稳定在 43 000 条左右;各周期内流的字节总数十分相近,数目稳定在 33 MB 左右。

b)各周期内各条流所占字节数差别很大,在 40 ~ 120 000 B 之间。

c)各周期内取不同的流阈值,所对应的流数目的变化情况十分相近,变化曲线大体为双曲线,如图 1 所示。

d)各周期内取不同的流阈值,所对应的流集合的字节数变化情况十分相近,变化曲线比较平滑,如图 2 所示。

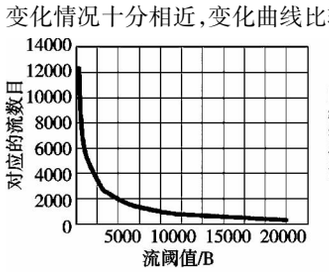

图1 流阈值—流数目变化图

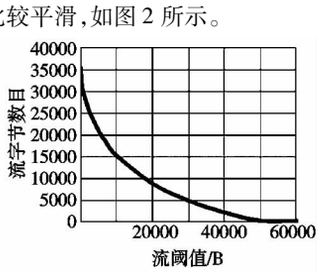

图2 流阈值—流字节数目变化图

2.2 较大流的选定

根据以上的结果可以推断,在周期内当取不同的流阈值,便得到一个流的集合,这个集合内流的数目及流字节数大体是稳定的。文献[6]作者便是取到一个流阈值,得到了一个流的集合,但是选取的这个阈值点比较特殊,使得集合内流数目占流总数的数目不多,但是字节比重比较大。对于本文采集的流量而言,这样的阈值点应至少满足以下两个条件:a)流阈值对应的流数目的值应该少于流总数的一半,大至在[40 B,21 000 B]区间内;b)流阈值对应的流字节数应该大于流字节总数的一半,大至在[17 M,33 M]区间内。同时满足这两个条件,流阈值的取值大至在[150 B,8 000 B]区间内。

本文延用了文献[6]对于较大流的定义。较大流是指在一定周期内所占字节数超过设定阈值的流,这些流具备流数目较少、但字节比重较大的特点。周期以处理一定数目的报文来计算。

通过综合考虑,选取了阈值 1 000 B 这个点,对应的较大流约占流总数的 13%,而所占的字节数比重达到 80% 以上。可以说网络数据流量对网络入侵检测系统的载荷压力主要来源于较大流,本文算法动态调整对象就是较大流,既解决了负载均衡问题,又尽可能减少了流的破坏。

2.3 HCLF 算法的思想

HCLF 算法框架如图 3 所示,分为三部分,即较大流识别、动态调整和静态哈希。算法中需要维护一张流状态表,在表中记录检测周期内所有流的流量,大于流量阈值的流即为较大流;系统中有多个检测引擎,以检测引擎的 CPU 利用率为负载量的指标,较大流通过负载均衡器被分配到某个检测引擎上,各检测引擎的 CPU 利用率信息按照周期反馈给负载均衡器。当检测引擎 CPU 利用率超过设定的负载阈值且到达的流属于较大流时,将该较大流分配到负载最轻的引擎上,而其他情况下,均按照静态哈希算法分发流。设定负载阈值的原因是 HCLF 算法是应用于入侵检测系统的算法,追求入侵检测系统的良好性能是本文研究的最终目标,负载均衡只是追求这一目标的手段。只要不影响系统的正常性能(如单核过载大量丢包),当三个检测引擎的负载均较低时,便没有必要使用动态调整进行负载均衡,这样可以节省系统资源,降低能耗。

定义检测引擎的 CPU 利用率函数为 $C_i(t)$,$i=1,2,\cdots,N$。该函数表示 t 时刻第 i 个引擎 CPU 利用率,该值越大,表示引擎的负载越重。

算法对网络流量的处理具有周期性,周期设定可以有两种思路,一种是按照每接收完一定量报文,另一种是按照一定的时间间隔(如 5 μs)。本文为了编程实现简单,减小开销,使用第一种思路,周期设定为接收 100 000 个报文。

3 系统实现

为了评估 HCLF 算法的有效性,本文基于图 3 的算法框架实现了原型系统。系统实现的硬件方面需要通用的、高性能多核平台,千兆网卡。笔者用两台高性能台式机进行实验:一台进行发包,硬件使用 Dell optiplex 755(双核 Intel® Pentium® Dual E2180/2 GHz,3 GB 内存,千兆网卡),软件包括 Linux Fedora10 平台上的 tcpreplay-3.4.4;另一台运行整个入侵检测系统,硬件使用 Dell xps8300(四核 Intel® Core™ i5-2300/2.8 GHz,4 GB 内存,千兆网卡),软件包括运行在 Linux Fedora13 平台上的 Snort-2.9.0.1[7] 和 PF_RING-4.6.0。

PF_RING 是 Luca Deri 提出的在高速网络环境下性能优秀

的报文捕获软件。通过设计一种 PF_RING 套接字,实现网络数据从网卡到用户空间的另一条途径,绕开了传统的 Linux 系统内核协议栈,通过内存映射技术减少了从内核空间到用户空间的数据拷贝。同时,由于 PF_RING 本身就是一种套接字协议,提供了对 socket 和 libpcap 的接口,使得其通用性和移植性都非常好,只需通过编译内核加载模块和打补丁,就能在各种类 Linux 系统上应用该技术[8]。PF_RING 的原理如图 4 所示。

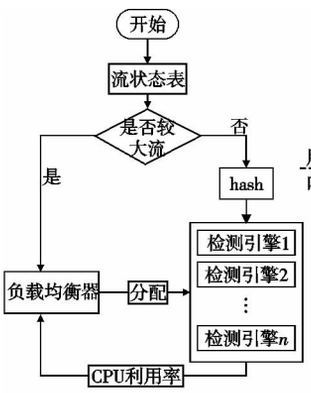

图3　HCLF算法示意图

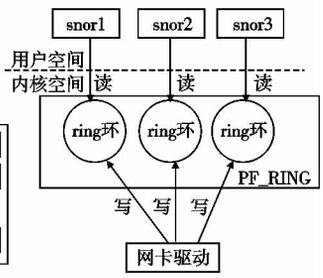

图4　PF_RING 工作原理

原型系统中硬件平台有四个处理核,在前三个处理核上每个核绑定并运行一个 snort 进程作为系统的检测引擎,每个 snort 进程使用一个 RING 环,网卡捕获的报文分配到三个 RING 环,应用程序 snort 从各自对应的 RING 环上接收报文并进行处理,第四核处理所有报文达到时产生的软中断。

关于 HCLF 算法的基本思想,已经在 2.3 节中介绍过了,在系统实现中的难点在于如何实时地将三个检测引擎的 CPU 利用率反馈给负载均衡器。由于算法的实现环境是在内核下,内核编程受到很多限制:不能使用用户态下常规的编程函数;不能使用 C 函数库;有些代码段由于加了读写锁不能在其中调用自己编写的函数;不能直接使用除法;内核态编程不支持小数运算等。有的方法能够实现需求,但是消耗系统资源严重,因而也不能使用。通过反复调试改进,最终采用如下方法:在加载 PF_RING 模块的同时利用 kthread_run 函数启用一个特定的线程,该线程运行笔者编写的计算三个检测引擎 CPU 利用率的函数,该函数将计算出的结果写入三个全局变量,并以一定时间进行刷新(系统中为 0.1 s),系统需要判断各个处理核的负载时,通过读取这三个全局变量的值来实现。但是这个线程是全程伴随整个系统运行的,维护这个特定的线程需要消耗某个处理核 3% 的 CPU 利用率。

4　实验结果与性能评估

4.1　评价指标

负载均衡算法首先需要关注负载均衡度。所谓负载均衡度是指网络流量在处理节点之间的分配量比值与处理节点之间处理能力比例的差异程度。负载均衡度越小,负载均衡性越好。在本文中处理节点(即检测引擎)的处理能力相等,所以负载均衡度就是各检测引擎负载量的差异程度。参考在文献[9]中定义的位流负载均衡度和报文负载均衡度,本文提出了 CPU 负载均衡度的公式:

$$cLBM(t) = 1 - \frac{(\sum_{i=1}^{m} C_i(t))^2}{m\sum_{i=1}^{m} C_i^2(t)} \tag{1}$$

其中:$C_i(t)$ 表示 t 时刻第 i 个引擎 CPU 利用率,m 为检测引擎的数目。

入侵检测系统性能有两个重要的衡量指标:误报率和漏报率。其中误报率与规则构成等因素有关,与负载均衡无关,不是本文关注点;漏报率与丢包率、规则构成等因素有关,其中丢包率对入侵检测系统性能影响非常直接,出现丢包,报文信息未被捕获,从而不会有相应的报警信息产生。

丢包率是指系统正常运行过程中丢弃的报文数与报文总数的比值,设 $D(t)$ 为 t 时间内丢弃的报文数,$P(t)$ 为 t 时间内检测节点灌输给入侵检测系统的报文总数,则 t 时间内丢包率 $Q(t)$ 为

$$Q(t) = \frac{D(t)}{P(t)} \tag{2}$$

基于多核平台的高速入侵检测系统中,在相同硬件环境、相同发包速率下,较低的丢包率能够直观体现负载均衡算法的优越性。

4.2　实验过程及结果

在实际应用中,入侵检测系统通常用来监测出入某一特定网段的流量,因此本文从采集到真实流量中提取了出入 AAA.0.0.0 ~ AAA.255.255.255 网段(真实地址隐去)的数据流量进行实验。

实验中用静态哈希算法与 HCLF 算法进行对比,静态哈希算法采用目的地址分段相加取模的方法。实验结果如图 5、6 所示。

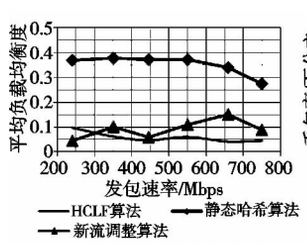

图5　平均负载均衡度对比

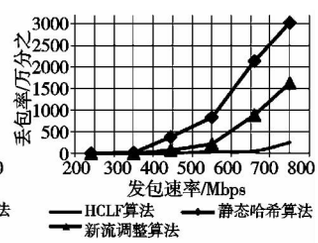

图6　丢包率对比

从图 5、6 中可以看出,HCLF 算法无论在丢包率还是负载均衡度方面均比静态哈希算法有显著优越性。

静态哈希算法自身优点明显,算法实现简单,执行高效,但是缺点也显而易见,它只是被动地按照既定的方法将各报文分流到各检测引擎。当网络数据中的报文按照这种方法不能够均衡地分配到各检测引擎,便会发生负载不均衡;当某检测引擎接收的报文数量超过其处理能力,即使其他检测引擎仍有处理能力,也无法实现分流,就会出现丢包。而对于 HCLF 算法,当网络数据对于各个检测引擎的负载量没有超过阈值时,仍实行静态哈希算法分流;当负载不均衡时且有处理核的 CPU 利用率超过阈值,本文设定 80%(通过实验,单个检测引擎的 CPU 利用率超过 80% 系统开始丢包)便实施动态调整,将应该分往该引擎的较大流调整到负载最轻的引擎,从而实现负载均衡。

HCLF 算法体现出优越性,但是这种算法的实现不可避免地带来了一定数量的流的破坏。在文献[3,5]中算法动态调整的对象是新到来的流,从而可以避免对流的破坏。本文按照调整新流的思路设计算法并实现了一种原型系统,对其进行测试,在图 5、6 中对应的为带三角的曲线,可以看出 HCLF 算法在丢包率和负载均衡度方面仍好于新流调整算法。

图 7 统计了在不同发包速率下,HCLF 算法和新流调整算法周期内动态调整的流数目(多个周期流数目的平均值)。

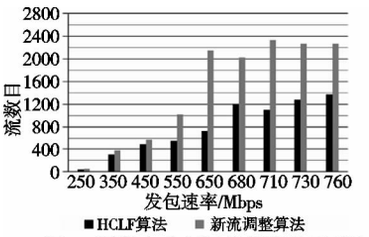

图7 周期内动态调整的流数目对比

图 7 中新流调整算法比 HCLF 算法在周期内调整的流数目要多一些,但也仅有一两倍的差距。需要注意的是:

a)新流调整的对象是任意流,而 HCLF 算法调整的流是较大流,较大流数目虽少,但流量比重占很大。如果将新流调整算法调整的流数目中的较大流数目提取出来,按照 2.1 节中的统计规律,其数目只有现在数目的约 13%;再将非较大流数目按照字节比重换算为较大流数,数目也只有现在数目的约 4%,两者相加,约为现在数目的 17%,与 HCLF 算法调整的较大流数目作比较,新流调整算法对应的数值会低于 HCLF 算法。

b)新流调整的对象是新到来的流,而周期内新到的流的数目是有限的,在负载不均衡的情况下,如果没有新流的到达,则不能进行动态调整。这说明新流调整算法的动态调整力度受到了算法本身的限制,而 HCLF 算法不受这个限制,能够充分发挥每个检测引擎的处理能力,最大化整个系统的处理性能。

综合以上两点,在相同硬件环境、相同发包速率下,HCLF 算法动态调整能力要强于新流调整算法,实验结果也证明了这一点。

5 结束语

本文提出了针对多核平台的高速入侵检测系统,以处理核 CPU 利用率为依据,动态调整较大流的 HCLF 算法。该算法能动态调整数据流量的分配,使各处理核达到较好的负载均衡,从而避免单核过载出现大量丢包。本文首先分析了骨干链路的真实流量特性;然后阐述了 HCLF 算法的思想和实现,通过实验,与静态哈希算法作比较,说明了 HCLF 算法的优越性,与新流调整算法作比较,说明了 HCLF 算法虽然带来了一定数量的流的破坏。但是由于其动态调整力度不受算法本身的限制,所以负载均衡度与丢包率仍优于新流调整算法。

参考文献:

[1] 程光,龚俭,丁伟,等. 面向 IP 流测量的哈希算法研究[J]. 软件学报,2005,16(5):625-658.

[2] 刘许刚,马宏. IP 流检测中基于信息熵的哈希算法改进[J]. 计算机工程,2011,37(16):94-97.

[3] 余颖,杨频,梁刚. 并行入侵检测系统的预测负载均衡方法[J]. 计算机工程与设计,2011,32(8):2565-2568.

[4] SHI Wei-guang, MACGREGOR M H, GBURZYNSKI P. A novel load balancer for multiprocessor routers[EB/OL]. [2011-08-29]. http://www. cs. ualberta. ca/macg/ Pub/2005/SPECTS-LOAD/FlowBal_I. pdf.

[5] 于洪伟. 基于多核处理器高效入侵检测技术研究与实现 [D]. 成都:电子科技大学,2009.

[6] 陆华彪,赵国鸿,陈一骄,等. 基于较大流调整的安全分流算法[J]. 计算机工程,2009,35(13):117-119.

[7] WALKERXK. 入侵检测:Snort-2. 9. 0. 1 发布[EB/OL]. (2010-11-03)[2011-09-20]. http://www. lupaworld. com/article-207556-1. html.

[8] 韩如冰. 千兆网环境下数据包捕获技术研究[D]. 武汉:华中科技大学, 2009.

[9] 陈一骄, 卢锡城,时向泉,等. 一种面向会话的自适应负载均衡算法[J]. 软件学报,2008,19(7):1828-1836.

(上接第 1412 页)抵御 DoS 攻击、放大效应攻击及欺骗攻击。最终实验结果表明,该算法对 IPv6 报头的安全性检测是一种有效的方法,它可以与安全防护设备联动,也可应用在中间设备和网关上。

4 结束语

为抵御互联网遭受基于 IPv6 报头格式的攻击,本文通过分析其存在的安全威胁,提出 IPv6 报头安全性检测算法。实验结果表明,该算法具备良好的检测性能,在为安全防护设备的过滤策略提供参考与建议的同时能增强安全防护设备对数据包全面检测的能力。

后续的研究工作应考虑:a)提高算法的检测效率;b)深入研究对片段报头与隧道封装机制结合的检测;c)展开检测算法与 IPSec 检测模块联动研究。

参考文献:

[1] CAICEDO C E, JOSHI J B D, TULADHAR S R. IPv6 security challenges[J]. IEEE Computer Society, 2009,42(2): 36-42.

[2] 陆音,石进,黄皓,等. 综述:关于 IPv6 安全性问题的研究[J]. 计算机科学,2006,33(5):5-11.

[3] FRANKEL S, GRAVEMAN R, PEARCE J, *et al*. Guidelines for the secure deployment of IPv6, SP 800-119[R]. [S. l.]: NIST, 2010.

[4] MONTGOMERY D, NIGHTINGALE S, FRANKEL S, *et al*. A profile for IPv6 in the U. S. government, SP500-267[R]. [S. l.]: NIST, 2008.

[5] CASIMIR A, POTYRA J. Firewall design considerations for IPv6, Report JHJ I733-041R-2007[R]. [S. l.]: NSA, 2007.

[6] 张玉军,田野. IPv6 协议安全问题研究[J]. 中国科学院研究生院学报,2005,22(1):30-37.

[7] RFC 2473, Generic packet tunneling in IPv6 specification[S]. [S. l.]: Network Working Group, 1998.

[8] RFC 2003, IP encapsulation within IP[S]. [S. l.]: Network Working Group, 1996.

[9] RFC 2460, Internet protocol version 6 (IPv6)[S]. [S. l.]: Network Working Group, 1998.

[10] RFC 2711, IPv6 router alert option[S]. [S. l.]: Network Working Group, 1999.

[11] RFC 4213, Basic transition mechanisms for IPv6 hosts and routers [S]. [S. l.]: Network Working Group, 2005.

[12] IPv6EDU. IPv6 协议漏洞:可能遭受 DDoS 攻击[EB/OL]. (2011-06-23)[2011-07-20]. http://ipv6edu. com/Html/4811. html.

[13] RFC 3775, Mobility support in IPv6[S]. [S. l.]: Network Working Group, 2004.