IPV6 环境下的高维大规模包匹配算法

王则林[1,2],吴志健[1],尹 兰[1]
(1.武汉大学计算机学院,湖北武汉 430072;2.南通大学计算机科学与技术学院,江苏南通 226019)

摘 要: 传统的包匹配算法不是无法运用于 IPV6 环境,就是性能太差.本文把基于实数编码的差分演化算法与传统的包匹配算法相融合.在适应值设计上引入变异系数的思想,从而使问题的处理更具有客观性.通过引入分布性特征,自适应调整变异的剧烈程度,从而动态权衡种群的多样性和收敛性之间的矛盾.数值实验表明此算法与传统算法相比,在速度、存储空间等综合性能上得到有效改善,另外本文提出的算法还有一个显著特点:包匹配的时间性能与规则数目之间具有很弱的相关性,从而本算法适合处理高维和大规模包匹配问题.本算法运用到 IPV6 网络,使数据包能快速转发.而且本文提出的方法具有普适性,适用于防火墙、路由器等网络设备.

关键词: 差分演化; 变异系数; IPV6
中图分类号: TP393 **文献标识码:** A **文章编号:** 0372-2112 (2013) 11-2181-06
电子学报 URL: http://www.ejournal.org.cn **DOI:** 10.3969/j.issn.0372-2112.2013.11.011

HigH-Dimension Large-Scale Packet Matching Algorithm in IPV6

WANG Ze-lin[1,2], WU Zhi-jian[1], YI Lan[1]
(1. *Computer School of Wuhan University, Wuhan, Hubei*, 430072, *China*; 2. *Computer Science and Technology of Nantong University, Nantong, Jiangsu* 226019, *China*)

Abstract: If the traditional packet matching algorithm is not able to be used in IPV6 environment, it must be of poor performance. This paper combines the real number coding differential evolution algorithm with the traditional packet matching algorithm, imports coefficient of variation in fitness value design and introduces distribution feature to adaptive adjust the acuteness degree of variation. Numerical experiments show that it has higher performance compared with the traditional ones. Package matching time performance has a weak correlation to the number of rule is another significant feature in the algorithm, so it is suitable for processing high peacekeeping large-scale package matching problem. Applying the algorithm to IPV6 network, the packet can linear speed forward. The proposed method is applicable to the firewall, router and other network equipment.

Key words: differential evolutionary algorithm; coefficient of variation; IPV6

1 引言

包匹配是下一代互联网络设备和新型网络服务的核心技术.对于包的匹配已有许多学者进行了研究,并提出很多重要的算法,但这些算法都不能很好的解决以最小包长和最小包间隔在路由器端口上双向传输的同时不引起丢包,即线速转发的问题.特别是 IPV6 逐步推广,传统的包匹配算法不是不适应新的要求,就是性能急剧下降[1].

目前研究的报文匹配算法可以概括成五类:基于 Trie 树的算法、基于空间分割的算法、启发式算法以及基于硬件的算法以及基于智能的算法.基于 Trie 的算法主要包括 Hierarchical Tires[2], Set-pruning Trie[3], Grid-of-Trie[4]以及 EGT-PC[5], Hierarchical Tires 的时间性能太好,Set-pruning Trie 是对 Hierarchical Tires 的改进,但空间消耗大大增加,Grid-of-Trie 是对 Set-pruning Trie 的进一步改进,时间空间性能都得到改善,但也无法做到在路由器、防火墙中报文的线速匹配,而且它只适用二维空间匹配,也不适用于 IPV6 环境.并且在初始数据结构的建立以及规则删除和插入时需要花费很长的时间.不适合运用于高维的情况,.基于空间分割的算法主要有 AQT[6], FTS tree[7], HiCuts[8]以及 HyperCuts[9]算法等,这些算法在时间效率方面在一些实际情况下优于基于 Trie 树的算法,但是只适用于低维度的情况,当维数变

收稿日期:2012-01-15;修回日期:2013-01-23;责任编辑:梅志强
基金项目:国家自然科学基金(No.61070008);教育部人文社科项目(No.11YJC870012);教育部人文社科基金(No.12YJCZH274);江苏省自然科学基金(No.BK2010277)

高,规则数增大时,会造成时间和空间复杂度的急剧增大,不适合 IPV6 高维的数据包匹配.HyperCuts 方式在现在的包匹配中常用.基于启发式的算法以 RFC[10]为代表,此方法在时间效率上目前是比较理想,但内存消耗非常大.在 IPV6 环境很容易发生空间爆炸.基于硬件的方法目前大多使用 TCAM[11],此方法虽然简单、高速,但价格昂贵、功耗非常大.

基于智能的算法有基于 ACO 的算法[12],解决组合优化问题的优势没有在此算法中得到充分体现,此算法随规则数的增长内存消耗显著增长,搜索速度也显著下降,不适合 IPV6 下的高维、大规模的数据包匹配.基于 DE 的算法[13]运用格雷码,在处理实数问题时存在明显缺陷,在分布性方面的考虑欠科学以及一些重要参数的选择缺乏动态考虑,没有考虑 IPV6 环境、考虑问题规模不大.本文首先分析研究解决实数编码问题的差分演化算法,然后将它与传统的包匹配算法进行融合,从而设计出一种基于实数编码的差分演化的包匹配算法(Real-based Differential Evolution Patcket Matching, RDEPM).

2 基本思想

数据包采用五元组(源 IP 地址、目的 IP 地址、源端口、目的端口、协议)来确定一个分组.在 IPV6 协议里 IP 地址以××××:××××:××××:××××:××××:××××:××××:××××:的形式存在.

源 IP 和目的 IP 地址从高位到低位,把每一段分别标识为 x_{ij}, $i\in\{1,2\}$, $j\in\{1,2,3,4,5,6,7,8\}$.

$$X_i=\text{Extract(IP)}=\sum_{j=1}^{8}(x_{ij} \bmod 1024\lambda),\ i\in\{1,2\} \quad (1)$$

其中 λ 参数的值根据网络规模进行设计,在处理 x_{i1}时,本算法只考虑单播的情况,最高三位的值固定为 001,故 x_{i1}不考虑最高三维的值.

源端口和目的端口都是以十六位二进制表示,以相应的十进制 Y_3、Y_4 标识. $X_i=Y_i \bmod 1024\lambda$, $i\in(3,4)$,上层传输层协议用八位的二进制表示,以相应的十进制 X_5 标识,设 $\boldsymbol{X}$ 表示向量(X_1,X_2,X_3,X_4,X_5).

$$F(X)=\left(\sum_{i=1}^{5}\alpha_i X_i+\lambda\beta\right)\bmod\ \ 1024\lambda \quad (2)$$

$F(\boldsymbol{X})\in(0,1024\lambda)$,并且 $F(\boldsymbol{X})$为整数,$0\leqq\alpha_i\leqq1$,$0\leqq\beta\leqq1024$.

依据公式(1)、(2),$F(\boldsymbol{X})$函数把 $\boldsymbol{X}$ 映射到一维空间$(0,1024\lambda)$. $F(\boldsymbol{X})$函数的目的就是把无规则的规则库映射到一个区间.

本文在映射函数 $F(\boldsymbol{X})$, $\boldsymbol{X}$ 域空间和 $F(\boldsymbol{X})$映射空间已知的条件下,识别 α_i 和β 的参数值.定义向量 $\boldsymbol{A}$ 为$(\alpha_1,\alpha_2,\alpha_3,\alpha_4,\alpha_5)$,本文运用差分演化算法来搜索使适应值最小的 $\boldsymbol{A}$ 和β 的组合.

本算法采用实数进行编码,以便引入与问题领域相关的启发式信息以增加演化算法的搜索能力.另外如运用二进制编码或 gray 编码,在 IPV6 环境将造成个体编码太长.在采用实数编码时要考虑下面两个问题.

(1)群体个体的描述:个体是一个实数向量,向量中的每个元素都是一个连续的变量.例如个体用向量$\boldsymbol{S}_i(s_{i1}\cdots s_{ij}\cdots s_{in})$, n 是问题的维数, $\boldsymbol{S}_i$ 表示第 i 个个体, s_{ij}表示第 i 个个体的第 j 个分量, s_{ij}是一个浮点数,范围在$[l(j),u(j)]$.

(2)在对个体进行变异操作时,如 s_{ij}的值超出$[l(j),u(j)]$,要进行相应的操作.可以选择按下列公式进行:

$$s_{ij}=\begin{cases}l(j), & s_{ij}<l(j)\\ u(j), & s_{ij}>=u(j)\end{cases} \quad (3)$$

也可以按下列公式进行

$$s_{ij}=\begin{cases}2l(j)-s_{ij}, & s_{ij}<l(j)\\ 2u(j)-s_{ij}, & s_{ij}>=u(j)\end{cases} \quad (4)$$

本文算法采用下列公式进行:

$$s_{ij}=\begin{cases}l(j), & s_{ij}<l(j)\text{and } r<0.5\\ 2l(j)-s_{ij}, & s_{ij}<l(j)\text{and } r>=0.5\\ u(j), & s_{ij}>u(j)\text{and } r<0.5\\ 2u(i)-s_{ij}, & s_{ij}>u(i)\text{and } r<0.5\end{cases} \quad (5)$$

其中 r 是一个范围在[0,1]之间的随机数.

基本差分演化算法:父代两个相异随机个体进行差操作得到的差分矢量加到随机选择的第三个相异个体上,生成一变异个体,接着按照一定的概率,父代个体与变异个体之间进行交叉操作,生成一新个体,在父代个体与新个体之间根据适应值的大小进行选择操作,选择适应值较优的个体作为子代.

本文变异时依据下列公式(6)进行操作.

$$\nu_m(t+1)=S_{\text{gbest}}(t)+\Delta(t,S_{r2}(t)-S_{r3}(t)) \quad (6)$$

其中 $m\neq\text{gbest}\neq r_2\neq r_3$, t 为当前演化代数,而函数$\Delta(t,x)$的值域为$[0,x]$或$[x,0]$,并使得当 t 增大时,$\Delta(t,x)$接近于 0 的概率大增,即 t 的值越大,$\Delta(t,x)$取值接近于0的概率越大,从而可以做到演化初期更多的考虑全局搜索,而在后期偏向于局部搜索.

$$\Delta(t,x)=x(1-\tau^{\varepsilon}),\text{其中 }\varepsilon=(1-t/T)^{\eta} \quad (7)$$

其中 τ 是[0,1]上的一个随机数, T 表示最大代数, η 是决定变异激烈程度的一个参数,起着调整局部搜索区域的作用,其取值一般为{2,3,4,5},本算法数值实验中此参数取值 2.

当 $\nu_{mj}(t+1)$ 超出$[l(j),\ u(j)]$,根据公式(5)进行处理. $S_{r2}(t)$, $S_{r3}(t)$是父代随机选择的两对个体,进行锦标赛选择得到的两个相异个体, $S_{\text{gbest}}(t)$是父种群最

优个体.

交叉时依据式(8)进行操作.

$$\mu_{mj}(t+1)=\begin{cases}\nu_{mj},(t+1), & \text{rand}()\leqslant C_R\\ s_{mj},(t), & \text{rand}()>C_R\end{cases}\qquad j=1,2,\cdots N. \tag{8}$$

$$C_R=\text{RED}_N/(64\lambda) \tag{9}$$

其中 RED_N 定义为:如 $\boldsymbol{X}$ 域中数目超过 M 阀值(本文中此值设置为 5)的点,且这些点从 $\boldsymbol{X}$ 域映射到 $F(\boldsymbol{X})$域同一个点,$F(\boldsymbol{X})$域中这样的点定义为冗余点,$F(\boldsymbol{X})$域中冗余点的数目定义为 RED_N.而 λ 值根据网络规模进行设置,λ 的值的设置确保 C_R 大于 1 为小概率事件.

如何做到 C_R 的值大于 1 是小概率事件,λ 必须如下设置:设规则数为 n,冗余点所映射掉的规则数为 S,被映射的区域离散点的规模为 Y(本文中设置为规则数规模 n 的 1.5 倍,即 $1.5n$),如果 $C_R>=1$,即 $\text{RED}_N>=64\lambda$,即 $S>=64M/\lambda$,从而要使 C_R 的值大于 1 是小概率事件,λ 设置时必须使$[(Y-S)/Y]^n$ 为小概率事件.

C_R 值越高,则 $\nu_m(t+1)$对 $\mu_m(t+1)$的贡献越多,有利于局部搜索和加速收敛速率;如果 C_R 越小,则 $\nu_m(t+1)$对 $\mu_m(t+1)$的贡献越小,有利于保持种群的多样性和全局搜索.由此可见,在保持种群多样性与收敛速率之间是矛盾的.在本文中 C_R 动态变化,而不是固定不变,在 $F(\boldsymbol{X})$域中被映射的点分布性较好时,N 的值会相应的降低,从而达到 C_R 的值自动变小,相反,N 的值会提高,从而 C_R 的值会相应增加.这样就能自动权衡种群多样性和收敛速度之间的矛盾.C_R 如设置一个恒定值,会带来预处理时间的增加、匹配平均时间的恶化,如表 1 数值实验所示:

表 1　C_R 不同设置带的包匹配性能比较

规则数	1K		5K		10K	
	预处理时间(s)	包匹配时间(s)	预处理时间(s)	包匹配时间(s)	预处理时间(s)	包匹配时间(s)
$C_R=0.5$	541	0.0 00002	1318	0.0000224	3721	0.0000028
$C_R=0.1$	1459	0.000 0037	3592	0.0000058	7815	0.0000084
$C_R=0.9$	1473	0.000 0036	3577	0.0000062	8264	0.0000079
C_R 动态	317	0.0000017	918	0.0000021	2003	0.0000022

选择操作依据下列公式(10)进行操作.

$$s_m(t+1)=\begin{cases}\mu m(t+1), & \text{if } \mu m(t+1)\text{适应值小于 } s_m(t)\\ s_m(t), & \text{otherwise}\end{cases} \tag{10}$$

在评价种群个体时,产生的个体是优还是劣,需要适应性度量.适应值计算方法设计的成功与否是一件关键性的工作,因为它是算法演化过程的驱动力,是自然选择的唯一依据,改变种群内部结构的操作皆通过适应值加以控制.

设计适应性度量:

$F(\boldsymbol{X})$域中冗余点的数目越少,即 RED_N 点越小,种群个体就越优.统计 $F(\boldsymbol{X})$域中被 $\boldsymbol{X}$ 域映射到的点,这些点的平均值记为 $\bar{E}$.再计算出标准差 S,根据 $C_V=S/\bar{E}$,计算出变异系数 C_V,$-C_V$ 值越小,种群个体就越优.

RED_N 和 C_V 都是从 $F(\boldsymbol{X})$域中被映射到的点的分布性能来考虑问题.利用统计学原理来保证点的分布性,分布性越好,内存空间占用就越少.而且包的匹配速度与 $F(\boldsymbol{X})$域中被映射到的点的分布性有很大关系,分布性性越好,包的匹配速度相应提高.

设 $\boldsymbol{x}$ 域映射到 $F(\boldsymbol{X})$中同一个点 i 的数目定义为 $H(i)$;V 为 x 域所有映射点的总和,定义如式(12)

$$V=\sum_{i=1}^{n}H(i) \tag{11}$$

其中 if　$H(i)=0$　then　$H(i)=\omega$,ω 可以调节,$F(\boldsymbol{X})$中如存在点没有被映射到会带来内存的浪费,适当把 ω 动态增加,得到的 $\boldsymbol{A}$,β 组合会减少内存消耗.本文的数值实验中取值为 1.

$$\text{令 } T(i)=\begin{cases}1, & \text{if } H(i)\neq 0\\ 0, & \text{Otherwise}\end{cases} \tag{12}$$

$$\text{令 } T=\sum_{i=1}^{n}T(i);\ V_{\text{AVE}}=V/T \tag{13}$$

由式(11)~(13)得到的 V_{AVE}这个指标可以用来分析 $F(\boldsymbol{X})$域中点被映射的密集度,通过它观察包匹配的时间效率随规则数增长的变化情况.它的值越小意味着被映射的点分布越均匀,分布性能越好.

$$\text{令 } f(\boldsymbol{A},\beta)=\phi\text{RED}_N-\psi C_V+\upsilon V_{\text{AVE}} \tag{14}$$

ϕ,ψ,υ 是用来调节 RED_N、G 以及 υ 对适应值的贡献大小,ϕ、ψ、υ 的值在 0 和 1 之间,和为 1,本算法取初始值为 0.3、0.3、0.4,如果在实验结果中观察发现,虽然适应值较优,但冗余点太多,此时就提高 ϕ 以及 υ 值,从而相应的降低 ψ 的值.衡量种群个体的指标就是 $f(\boldsymbol{A},\beta)$,它的值越小,个体就越优,寻找 $\boldsymbol{A}$,和 β 的最优组合,就是搜索使 $f(\boldsymbol{A},\beta)$的值最小的 $\boldsymbol{A}$,β 组合.

由式(9)中 RED_N 的描述知 $F(\boldsymbol{X})$域中的点可能有多个 $\boldsymbol{X}$ 映射,这样就可能有多个规则存放在同一个点,可以借助传统方法设计一个指针链表串联多个规则,以保证包匹配顺利进行.

当防火墙或者路由器收到数据包时,分析数据包的协议头,提取出 IP 地址,根据 Extract(IP)和 $F(\boldsymbol{X})$两函数计算出此数据包映射的点,如此点存在多个规则,依据指针链表进行顺序查找.

本文的基本思想运用差分演化算法预计算出的

$\boldsymbol{A}$、β 值,从而使散列函数的生成更具有科学性,然后结合指针链表组建规则表.

3 算法设计

由算法 1 产生 $\boldsymbol{A}$,β 最优的组合,根据 $F(\boldsymbol{X})$函数把所有规则映射到 0 到 1024λ 的一维空间,一维空间的每个点最多可能有 S 个规则,对这个规则设计一个指针链表串联.

算法 1

Input:最大代数 T,种群数 p_n,新旧个体差值的阀值 $\Delta S,\lambda,\phi,\psi,\upsilon,\eta,\omega$

Output:$\boldsymbol{A}$,β

Step1 初始化 $P(0)$,$t=0$;设置计数器 C 的初值为 0;

Step2 根据公式(14)构造出 f 函数计算 $P(t)$中每个个体的适应值,确定新的最优个体 $S'_{\text{gbest}}(t)$,计算和原来的最优个体 $S_{\text{gbest}}(t)$的差值,并把差值和阀值 ΔS 相比较,如小于阀值 ΔS,计数器 C 增加 1,

Step3 如果 t 大于 T 或者计数器 C 的值大于 3,则停止算法运行,并输出最优解;否则转 step4;

Step4 根据锦标赛选择产生两个相异个体 S_{r2},S_{r3};

Step5 $P(t)$根据公式(6)进行变异产生 $\nu_m(t+1)$,根据公式(9)计算出 C_R;

Step6 根据公式(8)进行交叉操作产生 $\mu_m(t+1)$;

Step7 根据公式(10)进行选择操作产生 $S_m(t+1)$;

Step8 $t=t+1$,转 **Step.**

当一个数据包到达防火墙、或者路由器时,执行算法 2.

算法 2

Input:$\boldsymbol{A}$,β,到达的数据包

Output:对包进行相应的处理

Step1 根据到达的数据包,提取出 IP 地址,根据 Extract(IP)函数等,计算出 $\boldsymbol{X}$ 向量的值,再根据 algorithm(1)演化出的 $\boldsymbol{A}$ 和 β 组合,代入 $F(\boldsymbol{X})$中;

Step2 根据 $F(\boldsymbol{X})$映射的一维空间的相关点,搜索此点对应的规则,如超过一条规则,则根据传统的顺序查找的算法在链表中进行搜索;

Step3 如果规则不存在,依据相应的缺省规则进行处理;

Step4 如规则存在,以规则中规定的动作进行相应处理.

4 算法分析

4.1 时间复杂性分析:

定理 本文提出的 RDEPM 算法能够实现 IPV6 环境下的高维大规模数据包的快速转发.

证明

(1)算法 2 第一步根据式(1)易知,Extract(IP)的时间复杂度为 $\Theta(1)$;即算法 2 中 Step1 的时间复杂度为 $\Theta(1)$

(2)算法 2 中 Step2 由 $\boldsymbol{X}$ 计算出 $F(\boldsymbol{X})$的时间复杂度为 $\Theta(1)$,即算法 2 中 Step2 的时间复杂性为 $\Theta(1)$.

(3)算法 2 中 Step3 由算法 1 易知,$\boldsymbol{X}$ 域映射到 $F(\boldsymbol{X})$域中同一点的 $\boldsymbol{X}$ 数目不可能超过阀值 M(本文验证实验中设为 5),在链表中查找匹配具体规则时,成功搜索的平均时间消耗为 $M/2$,最坏情况下为 M,搜索不成功的时间消耗为 M,无论查找匹配成功与否,无论最坏情况,还是平均情况,时间消耗都是 $\Theta(1)$,即算法 2 中 Step3、Step4 的时间复杂度为 $\Theta(1)$.

(4)$\Theta(1)+\Theta(1)+\Theta(1)$的时间复杂度为 $\Theta(1)$,即算法 2 的时间复杂度为 $\Theta(1)$

从而得证 RDEPM 算法能够快速转发数据包.

在生成规则库的过程中,即预处理过程相对于传统的方式还是耗时略多.但对于规则的删除以及在链表中的规则数不超过阀值的条件下的规则添加也会做到线性时间消耗.

4.2 空间复杂性分析:

内存空间消耗在数据包转发过程中,是 $\Theta(1)$.但在数据预处理过程中为 $O(N)$,N 为规则库中规则数的规模.存储规则时需要消耗约 $1.5N\sim 2N$ 的范围的内存空间,一般为规则数的 1.5 倍,当 $F(\boldsymbol{X})$域中映射点的分布性能不良时,可以通过调节内存消耗量来改善,但因为本算法中 C_R 的动态变化在某种程度上减少了这种可能.另外在公式 11 中引入动态 ω,也是从改善内存消耗的角度考虑.

4.3 被映射点的分布性分析:

公式 C_R、C_V 以及 V_{AVE},都是从不同角度来衡量映射点的分布性,在算法(1)中限制 $F(\boldsymbol{X})$域单点被映射的次数 M,另外从适应值函数角度来选择映射点分布性较优的 $\boldsymbol{A}$ 和 β 组合.从而保证映射分布的良好性能.本文在计算适应值时还采用人机交互的模式,在适应值良好,但分布图效果明显存在冗余点过多时,调节适应值的 ϕ、ψ、υ 参数,适当降低 ψ,提高 ϕ 和 υ 的值.从而改善分布性能.

5 实验和结果

将本文提出的算法与算法 RFC 和 HyperCuts 进行对比实验,RFC 是目前包匹配速度最理想的算法,HyperCuts 是目前运用的比较多的算法,综合性能较优.实验是在模拟环境下进行的.在实验中,设置防火墙数据包每秒钟到达 400k 个,每次测试时间为 10s.仿真所用的操作系统平台是 RHEL5.0,CPU 是酷睿 E7500 双核 2.93HZ,4G 内存;所用的模拟器是 Network Simulator v2.27.基于 6 组实验数据,观察当规则数线性增长时,三种算法的各个性能参数的变化情况.本实验对象是防火墙规则.规则生成遵循随机的原则,实验结果是运算 60 次的均值.

表 2　3 种算法性能比较

	预处理时间(s)	每个包匹配时间(s)	内存消耗(MB)	预处理时间(s)	包匹配时间(s)	内存消耗(MB)
		规则数 500			规则数 1k	
RDEPM	134	0.0000015	23.46	317	0.0000017	35.41
RFC	763	0.0000012	27.52	361	0.0000014	95.17
HperCuts	23	0.0000122	40.23	49	0.0000242	45.17
		规则数 3k			规则数 5k	
RDEPM	776	0.0000019	42.30	918	0.0000021	.73.52
RFC	782	0.0000027	403.26	1362	0.0000039	896.85
HperCuts	257	0.0000478	55.39	672	0.0000849	59.62
		规则数 8k			规则数 10k	
RDEPM	1573	0.00000021	95.92	2003	0.0000022	121.32
RFC	2134	0.00000061	2013.02	5263	0.0000085	3276.39
HperCuts	1023	0.00001241	68.97	1429	0.0003178	83.02

表 2 展示了 3 种算法的数据结构建立时间消耗、数据包转发平均时间消耗、以及内存平均消耗.

从表 2 以及图 2 来看,本文提出的 RDEPM 算法在预处理时间消耗上比 RFC 优越;从表 2 以及图 3 可以发现,本文提出的 RDEPM 算法在内存消耗性能上远远优越于 RFC,而且从图 3 看,RFC 的内存消耗随着规则数的增加剧烈增加.这种内存消耗爆炸性增长方式不能适应大规模包匹配.

从表 2 和图 2 可以观察出,本文提出的算法 RDEPM 在数据包的匹配平均时间消耗上优越于 HiperCuts,更难能可贵的是,本文提出的 RDEPM 算法随规则数规模的增长,数据包匹配的平均时间消耗只呈现微弱的增长,而 HiperCuts 的增长幅度比较大.从而说明虽然 HiperCuts 在预处理以及内存消耗上有一定的优势,但由于数据包的匹配时间消耗性能欠佳,也不适合 IPV6 环境下的大规模包匹配.

从综合性能角度,本文提出的算法综合 RFC 的数据包匹配时间消耗性能的优越性和 HiperCuts 的内存消耗的性能的优越性.适用于 IPV6 环境下的大规模包匹配,本算法在 10k 的规则数规模下,能做到 454.545kpps.

从图 4 到图 9 可以观察到 X 域在 $F(X)$域上映射的分布情况,在规则数相应从 500 线性增长到 10k 时,

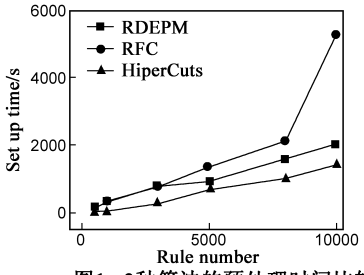

图1　3种算法的预处理时间比较

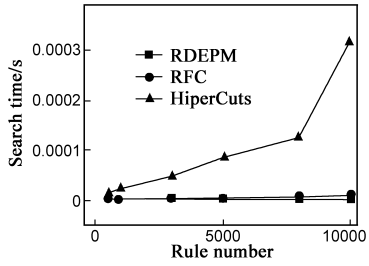

图2　3种算法每个数据包匹配的平均消耗时间比较

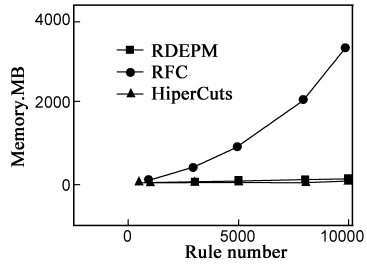

图3　3种算法的平均内存消耗比较

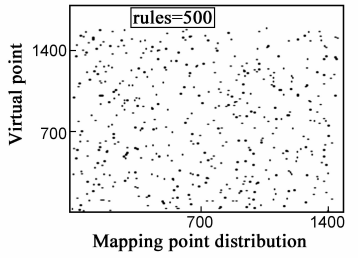

图4　规则数为500时的的映射点分布

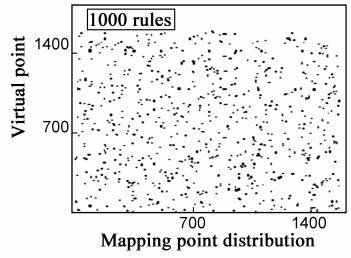

图5　规则数为1k时的映射点分布

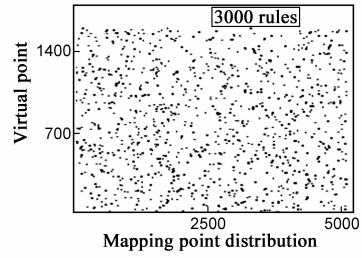

图6　规则数为3k时的映射点分布

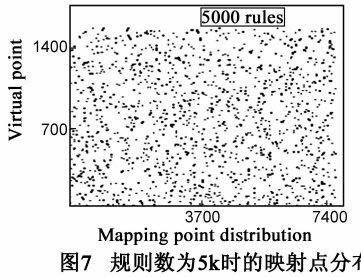

图7　规则数为5k时的映射点分布

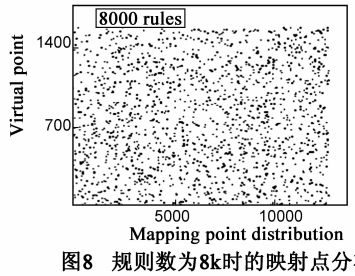

图8　规则数为8k时的映射点分布

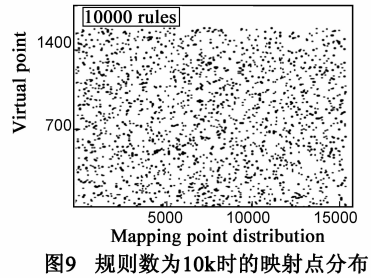

图9　规则数为10k时的映射点分布

发现分布效果理想,从而保证包匹配的平均性能相应很好.分布性理想意味着在 $F(X)$域上的冗余点比较少,这样就使链接的规则数超过阀值的链表数量较少.从而改善包匹配的时间消耗性能.

6 总结

本文把差分演化算法运用于 IPV6 环境下数据包的匹配,从而建设性地解决在合理的内存消耗下,实现数据包快速转发问题.本文提出的算法非常适合于 IPV6 环境下的高维大规模的数据包的匹配.由于处理的是 IPV6 环境下的高维大规模问题,本算法在预处理时间消耗性能上,相对于 HiperCuts 等经典算法还是存在一定的差距.本文的进一步工作就是运用演化算法本身的固有并行性,借助 GPU 进一步解决此问题;此外探索融合相应的智能算法来进一步改善算法的性能.

参考文献

[1] 李振强,郑东去,马严.一种多阶段 IPv6 路由表查找算法[J].电子学报,2007,35(10):1859 – 1864.
LI Zhen-qiang, ZHENG Dong-qu, MA Yan. A multi-stage algorithm for IPv6 routing table lookup[J]. Acta Eletronica Sinica, 2007, 35(10): 1859 – 1864. (in Chinese)

[2] P Warkhede, S Suri, G Varghese. Fast packet classification for two-dimensional conflict-free filters[A]. Proceedings of IEEE, INFOCOM, Twentieth Annual Joint Conference of the IEEE Computer and Communications Societies[C]. Alaska: IEEE, 2001.1434 – 1443.

[3] P Gupta, N McKeown. Algorithms for packet classification[J]. IEEE Network, 2001, 15(2): 24 – 32.

[4] V Srinivasan, G Varghese, S Suri, et al. Fast and scalable layer four switching[A]. Computer Communication Review[C]. Vancouver: ACM SIGCOMM, 1998.191 – 202.

[5] Buddhikot M M Suri S, Waldvogel M. Space decomposition techniques for fast layer-4 switching[A]. Proc of Conf On Protocols for High speed Networks[C]. Salem: IEEE, 1999.25 – 41.

[6] Feldman A, Muthukrishnan S. Tradeoffs for packet classification[A]. Proceedings of INFOCOMM, March[C]. Aviv, Israel: IEEE, 2000.1193 – 1202.

[7] Gupta P, McKeown N. Packet classification using hierarchical intelligent cuttings[J]. IEEE Micro, 2000, 20(1): 34 – 41.

[8] S Singh, F Baboescu, G Varghese, J Wang. Packet classification using multi-dimensional cutting[A]. Proceedings of The 2003 Conference on Applications, Technologies, Architectures, and Protocols for Computer Communications[C]. Karlsruhe: ACM SIGCOMM, 2003.213 – 224.

[9] Gupta P, Mckeown N. Packet classification on multiple fields[A]. Proc SIGCOMM, Computer Communication Review[C]. Massachusetts: ACM SIGCOMM, 1999.147 – 160.

[10] J van Lunteren , A P J Engbersen. Multi-field packet classification using ternary CAM[J]. Electronics Letters, 2002, 38(1): 21 – 23.

[11] 李维,刘斌,郗颖,等.基于多域并行编码的高速 IPV6 流分类[J].电子学报,2007,35(5):976 – 981.
LI Wei , LIU Bin, XI Ying, et al. Ultra-high speed IPv6 packet classification based on parallel multi-field encoding[J]. Acta Eletronica Sinica, 2007, 35(5): 976 – 981. (in Chinese)

[12] N K Sreelaja, G A Vijayalakshmi Pai. Ant colony optimization based approach for efficient packet filtering in firewall[J]. Applied Soft Computing, 2010, 10(4): 1222 – 1236.

[13] Zelin Wang, Zhijian Wu, Buzhong Zhang. Packet matching algorithm based on improving differential evolution[J]. Wuhan University journal of natural sciences, 2012, 17(5): 447 – 453.

作者简介

王则林 男,1973 年 12 月出生,江苏南通人.现为武汉大学计算机学院软件工程国家重点实验室博士生,从事智能计算,网络安全等方面的有关研究.

E-mail: whwzl@whu.edu.cn

吴志健 男,1963 年 8 月出生,江西鄱阳人,博士生导师,现为武汉大学软件工程国家重点实验室副主任,湖北省软件测试中心主任.主要研究领域为智能计算和并行计算.