一种双哈希IP数据包分类算法研究*)

尚凤军[1,2] 潘英俊[1]

(重庆大学光电工程学院 光电技术及系统教育部重点实验室 重庆400044)[1]

(重庆邮电学院计算机科学与技术学院 重庆400065)[2]

摘 要 本文在无冲突哈希算法和异或哈希算法的基础上,提出了一种双哈希的IP分类算法,该算法的核心有三点:一是基于目的/源端口和协议域构造无冲突哈希,由于该三域的组合数目非常少,避免了空间爆炸;二是在异或哈希算法的基础上,将目的/源IP连成比特串后分为四块后进行异或,为了降低冲突率,将异或后的关键值再与一个随机数进行异或,获得分类索引值,并用此值生成多比特Trie树,一般情况下减小了空间和时间复杂度;三是在Trie树终点存放最终分类规则的索引值,为了保证查找到的规则的正确性,对每一个索引值的源/目的IP地址均匹配一次。通过以上三点改进一般要降低算法的时间复杂度和空间复杂度,通过仿真,当对1万条分类规则进行包分类时,该算法的包分类速度可以达到2MPps,所消耗的最大内存为4MB。

关键词 IP分类,查找算法,多比特Trie树,双哈希

Research on a Double-Hash IP Classification Algorithm

SHANG Feng-Jun[1,2] PAN Ying-Jun[1]

(Key Laboratory of Opto-electronic Technology and System of Ministry of Education, College of Opto-electronic Engineering, Chongqing University, Chongqing 00044)[1]

(College of Computer Science and Technology, Chongqing University of Posts and Telecommunications, Chongqing 400065)[2]

Abstract In this paper, the authors survey the recent advances in the research of IP classification and introduce some of the typical algorithms. We describe a combination scheme that combines the advantages of both schemes. At last, a novel IP classification is proposed based on the double-hash algorithm, which is based on Non-Collision Hash Trie-Tree Algorithm and XOR hash algorithm. The core of algorithm consists of three parts: (1) structure the non-collision hash function, which is constructed mainly based on destination/source port and protocol type field so that the hash function usually can avoid space explosion problem; (2) introduce multibit Trie-tree based the key value of XOR hash in order to reduce time complexity; (3) lookup every rule index in order to ensure the validity that we get the final rule index. The test results show that the classification rate of double-hash algorithm is up to 2 million packets per second and the maximum memory consumed is 4MB for 10,000 rules.

Keywords IP classification, Lookup algorithm, Multibit trie-tree, Double hash

1 引言

随着网络的发展,客户终端要求网络服务提供商提供更有价值的服务,包括包过滤、流量计费和更好的服务质量等,所有这些都需要IP分类技术。

下一代通信网络是基于IP技术的综合话音、数据、图像、视频的多媒体业务网络,能够为不同的业务类型提供优良的、不同级别的服务质量,并且能够提供宽带接入能力。业务量测量是实施有效的网络管理和控制的前提,业务量测量结果是对网络运行状况优劣进行评判的依据。同时业务量测量对于网络容量的规划、按用户对网络资源占用情况实施计费的新措施的实现等都有非常重要的作用。业务量测量的关键技术之一便是流分类技术,由于流分类中处理的数据包绝大部分都是IP包,因此相应的技术也称为IP分类(IP classification)技术。

IP分类问题是最佳过滤规则匹配问题的一个实例,原则上讲,共有7个域可被选择成为过滤规则的域:源/目的IP,源/目的端口,TOS域,协议域和协议标志。实际上,过滤规则并不是对所有的域都感兴趣。统计发现,17%的过滤规则指定了1个域,23%的过滤规则指定3个域,60%的规则指定4个域,鉴于以上统计本文中选用5个域[1]。

IP分类问题的核心是高效的查找算法,衡量一个优秀查找算法的性能测度一般有以下四个:(1)时间复杂度。对该测度的研究一般指读取内存的次数,一般分为平均分类时间和最坏分类时间;(2)空间复杂度。一个算法的最终消耗的内存容量是一个非常重要的测度。一般空间复杂度越小越好,虽然目前内存的大小已经不是关键的问题,但若要求硬件完成分类算法,空间复杂度必须足够小,这样才能保证时间复杂度最低。(3)算法更新速度。如果分类规则不是静态的,则更新速度是一个非常重要的指标,一般越小越好。(4)可扩展性。一个好的分类算法,要求有非常好的扩展性,就目前而言,尚未有一个非常好的办法,至此,大部分分类算法均未提出一个合理的解决方案。目前对一个算法的评价主要采用时间和空间复杂度两个测度参数。

对于目前的数据包分类算法一般可以分为以下三类:(1)一维算法,主要应用于路由器中,以单层Trie树为代表。(2)

*)本文研究得到重庆市科技攻关重点项目(合同编号:7220-13-20)、重庆邮电学院青年教师基金资助(合同编号:A2003-03)和重庆市自然科学基金资助。尚凤军 讲师,博士生,研究方向为智能仪器、网络管理及测量。潘英俊 教授,博导。

二维算法，以 Grid of Tries 算法为代表[2]。该算法扩展了 Trie 树数据结构，用二级 Trie 树实现目的/源 IP 对的分类问题。对于完整的两层 Trie 树结构来说，空间浪费较大，在文[2]中采用了修剪二层 Trie 树结构使空间性能下降到 $O(NW)$（N 为规则数，W 为 IP 宽度），但对于修剪 Trie 树来说时间性能提高至 $O(W^2)$，于是文[2]中又引入了转向指针使时间复杂度下降到 $O(W)$。因此对于引入转向指针的 Trie 树查找算法来说，是一种优秀的二维 IP 分类算法，但它只适应于二维分类算法，对于多维分类算法来说时间复杂性较差。(3)多维算法，一般可分细分为四类：一类以统计学为基础，主要以 Modular 算法[5]为代表，它是一种基于统计的包分类的算法，根据对过滤规则的匹配率的分布和 IP 包的分布优化查找数据结构。但是，由于目前尚不能建立有效的统计参数，这种算法还难以用到实际的路由器中；第二类以递归方法为基础，主要以 RFC(Recursive Flow Classification[1])算法为代表，它是一种多维 IP 分类快速查找算法。其主要思想是将 IP 分类问题看成一个将包头中的 S 比特数据到 T 比特的 classID 的一个映射（T<<S，N 是过滤规则的总数）。它需要大量的预计算工作，容易发生空间爆炸，目前此方法是分类速度最快的一种算法；第三类以 Grid of Tries 算法为基础，主要是对该算法进行改进和扩充，如无冲突哈希查找算法[3]，基于跳转表的多维 IP 分类算法[4]等，在极端情况下，该算法均可能会产生内存爆炸；第四类以哈希算法为基础，主要以 Cross-Product 算法[3]为代表，Cross-Product 的 Cache 策略是基于对过滤规则中各域值的交叉组合的“缓存”，每一个交叉组合对应一类 IP 包，因此用少量的 Cross-Product 就可以代表相当数量的一类 IP 包。在 Cross-Product 中表现出的局部性也相应好得多，Cache 的效率也高得多，在极端情况下，该算法可能会产生内存爆炸。

本文讨论了一种基于双哈希和多比特 Trie 树的多阶段多维 IP 分类算法，该算法在时间和空间平均性能均优于 Grid of Tries 多维分类算法。

2 双哈希和多比特 Trie 树分类算法

2.1 算法基本思想

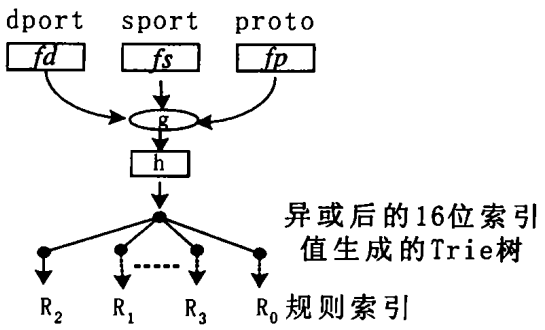

图1 双哈希算法结构示意图

本文讨论了一种双哈希的多阶段多维 IP 分类算法，该算法是在无冲突哈希和异或哈希算法的基础上提出的，其算法结构如图1所示（其中异或哈希部分未画出）。本文在无冲突哈希算法和异或哈希算法的基础上，提出了一种双哈希的 IP 分类算法，该算法的主要思想是通过两次哈希查找来最终确定数据包的分类，即属于哪一条规则，双哈希之间通过多比特 Trie 数进行连接。该算法的核心有三点：一是基于目的/源端口和协议域构造无冲突哈希，由于该三域的组合数目非常少，避免了空间爆炸；二是在异或哈希算法的基础上，将目的/源 IP 连成比特串后分为四块后进行异或，为了降低冲突率，将异或后的关键值再与一个随机数进行异或，获得分类索引值，并用此值生成多比特 Trie 树，Trie 树的终点存放相应规则的索引值，一般情况下减小了空间和时间复杂度。由于异或后一般情况下有冲突，因此 Trie 树终点可能有多个索引值，仿真时发现，对于10000条规则的情况下，一般有30个索引值，然后对每个规则进行线性查找即可；三是在 Trie 树终点存放最终分类规则的索引值，为了保证查找到的规则的正确度，对每一个索引值的源/目的 IP 地址均匹配一次。

未来 IP 地址向 IPv6方向发展，对于128位 IP 地址的包分类算法，采用哈希算法应该是比较理想的。本文讨论了一种基于 XOR Hash 的多维 IP 分类算法，该算法首先将包头数据连成比特串 h，然后分块后将 $h[2]$映射到另一个随机空间 g，最后将将映射到另一随机空间的随机数与剩下的三块进行异或运算得到规则的索引值 R，以此索引值作为生成 Trie 树的比特值。数据包到来后，将包头中源/目的 IP 对连接成比特串，分割为四块后读取某一块的随机值与剩下的三块进行异或运算后直接读取内存的索引值作为多比特 Trie 树的索引值，一次查找即可确定分类规则。

$h=\{h[1],h[2],h[3],h[4]\}$，其中 h 为源/目的 IP 对包头比特串，$f:h[2]\rightarrow g$，其中 f 为随机数发生器。

$R=h[1]\oplus h[3]\oplus h[4]\oplus g$，其中 R 即为异或后的多比特 Trie 树的索引值。

定理1 随机数发生器产生的随机数服从均匀分布，即 $g\sim U(0,m)$，其中 m 为随机数产生空间的上界。

定理2 随机数与常数异或运算后仍为随机数，即 R 仍服从均匀分布。

定理3 在均匀分布空间中，异或哈希算法的冲突率是有界的，下面简单加以说明：

对随机数集合$\{R_1,R_2,\cdots,R_m\}$，其空间大小为 m，因此每个元素出现的概率为$1/m$，从中随机抽取 x 个元素，出现最多 y 个冲突的概率可以表示为（P 表示排列，C 表示组合）：

$$(P_m^x+C_x^2P_m^{x-1}+C_x^3P_m^{x-2}+\cdots+C_x^yP_m^{x-y+1}+C_x^{y+1}P_m^{x-y})(1/m)^x$$

一般情况下，过滤规则基本上只限于5个域：目的/源 IP、目的/源端口，协议域，我们首先将目的/源端口和协议域构建成无冲哈希。然后将源/目的 IP 对连接成比特串，共64比特，接下来将这64比特平均分割为四个块，每块16比特，如图2所示，然后将某一16比特映射到另一随机空间，接下来将另一随机空间的随机数与剩下的三块进行异或运算得到16比特的 Trie 树索引值。最后通过查找获得分类规则索引集合，经过线性查找即可确定最终分类规则。

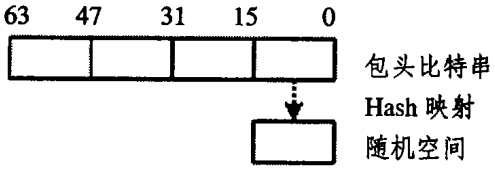

图2 源/目的 IP 对位串连接图

为了便于包分类处理，我们还需建立一个哈希桶，用于存储相应的随机空间中的16比特随机数，桶中为包头比特串映射后的16bit 的随机数。

对于异或哈希算法来说冲突的处理是非常关键的，本文中主要采用了如下几种方法降低冲突的次数：(1)对规则进行

分析处理，确保规则中无冗余规则，冲突规则最少；(2)采用随机数方法将规则随机映射到哈希桶中，这样能有效地减少冲突的规则数；(3)通过随机数方法将冲突大大降低(与直接异或比较)，对于1000条规则冲突数不超过5条。由于异或后的结果仍为随机数，因此冲突可以降到最低。由于两个随机数异或后仍为随机数，因此最终的冲突次数一定是有界的。

在实际工作中我们使用的都是伪随机数，它是介于一定范围之间的随机数，每一个数出现的几率都是一样的。理想情况下，应生成0到1之间的一个值，不考虑以前值，这个范围中的每一个值出现的几率都是一样的，然后再将该值乘以最大值，由于在0和1之间有无穷多个值，而计算机不能提供这样的精度，因此我们得到的是伪随机数。

为了编写代码来实现类似于前面提到的算法，常见情况下，伪随机数生成器生成0到N之间的一个整数，返回的整数再除以N。得出的数字总是处于0和1之间。对生成器随后的调用采用第一次运行产生的整数，并将它传给一个函数，以生成0到N之间的一个新整数，然后再将新整数除以N返回。这意味着，由任何伪随机数生成器返回的数目会受到0到N之间整数数目的限制。在大多数的常见随机数发生器中，N是2^{32}(大约等于40亿)，对于32位数字来说，这是最大的值。换句话说，我们经常碰到的这类生成器能够至多生成40亿个可能值。

在本分类算法中，随机数的产生是一个非常关键的部分，它直接影响到哈希冲突的次数，实验证明随机数的随机性越大，哈希分类算法的冲突次数就越低。

伪随机数是指用数学递推公式所产生的随机数。伪随机数产生的方法多种多样，其中最常用的有线性乘同余法、移位寄存器法、混合法和Fibonacci法。我们采用线性乘同余法，其基本形式是对任意初值整数X_0，由如下递推公式确定：

$$X_i=(aX_{i-1}+c)\bmod M \quad (1)$$

$$\xi_i=X_i/M, i=1,2,3,\cdots, \quad (2)$$

其中，乘子a为小于M的正整数，c为非负整数，常选为零；从实际经验看，当$M=2^{31}-1$，$a=16087$或630360016时得到的随机数ξ_i均匀性和独立性较好，因而得到广泛应用。

在实际工作中我们采用把线性乘同余法与*Visual C*$^{++}$的*srand*()和*rand*()两函数结合起来的方法。其做法是用*srand*()和*rand*()两函数来产生初值X_0，随后由(1)，(2)式就可得到随机变量系列ξ_i。其中ξ_i为0—1之间的有理数。

2.2 无冲突哈希查找

一般情况下，过滤规则基本上只限于5个域：目的/源IP、目的/源端口，协议域。实际上，目的端口和源端口的取值往往是0-65535中极少的一部分，协议域的取值是0—255中很少几个值，目前协议域实际上能够取的值只能是TCP、UDP、ICMP、IGMP、(E)IGRP、GRE和IPINIP。在Client-Server结构中，所有端口被分为两类：一类是保留端口，端口号为1—1023，另一类是临时端口，端口号为1024—65535。通常极少过滤规则会对客户端端口产生兴趣，一般是要求其大于1023[3]。在绝大多数情况下，保留端口通常是20，21(FTP)和80(WWW)，其它端口很少用。因此实际的过滤规则端口和协议域的取值(或取值范围)的不同情况的组合数目是非常有限的。基于此，我们建立了一个表，表中存放交叉组合后的索引值，由于端口和协议域取值非常有限，因此表的大小也很有限。下面以表1为例来说明具体操作过程：

表1 简单规则集

ClassID	目的IP	源IP	目的端口	源端口	协议
0	10.1.*.*	10.2.*.*	*	*	*
1	10.3.*.*	10.4.*.*	80	*	17
2	10.5.*.*	10.6.*.*	80	*	17
3	10.7.*.*	10.8.*.*	[20,21]	*	6
4	10.9.*.*	10.*.*.*	*	gt1023	6
5	*	*	*	*	*

gt1023表示源端口号大于1023的源端口。

算法1 *fd*表的建立(*fp*，*fs*表建立方法相同)

(1)根据规则库中所有目的端口值组成目的端口集合，然后为该集合中的元素进行标号。例如21是0，80是1，90是2，直到所有的目的端口都标完号为止(最坏情况下最大标号为65535)；

(2)当有新的规则加入时，可以直接追加到表的末尾即可。查表时，表的地址即为表项的内容。

算法2 *h*表的建立

(1)当*fd*表、*fs*表和*fp*表建立好之后，就可以根据我们构建的无冲突哈希函数生成表*h*；

(2)当有新的规则加入时，要重新生成*h*表，一般情况下*h*表很小，故重新生成时间一般也很短(一般为ms级)。

对于一个包头$H(dport, sport, proto)$(分别表示目的/源端口和协议域，此时暂时不考虑两个IP地址)，分别以*dport*、*sport*和*proto*为索引查表，得到$fd(dport)$、$fs(sport)$和$fp(proto)$，然后选取以某个函数$g(fd(dport), fs(sport), fp(ptoto))$为索引再进行查表，得到$h(g(fd(dport), fs(sport), fp(proto)))$，此数值即为多比特Trie树集合的指针。对所有目的端口D以0，1，……，$D-1$依次编号(D为端口总数，最大为65535)，同理对协议域P和源端口S也进行编号(P最大为255，S最大值为65535)。此过程如图3所示。方框表示查表，g是哈希函数，选择$g(d,s,p)=PSd+Ps+p$，其中，$0\leqslant d\leqslant D-1$，$0\leqslant p\leqslant P-1$，$0\leqslant s\leqslant S-1$，很显然哈希函数$g$是无冲突的[3]。

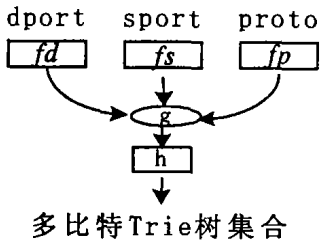

图3 目的/源端口和协议域分类示意图

这样根据源/目的端口和协议域的值一次确定多比特Trie树集合的指针，接下来，便可以查找基于异或哈希获得的多比特Trie树，进而完成分类工作。

2.3 源和目的IP对分类算法

在这一部分中，我们在异或哈希算法的基础上提出了一种多比特Trie树的查找算法。首先我们将目的/源IP地址连成比特串，共64比特，接下来将这64比特平均分割为四个块，每块16比特，然后将某一16比特映射到另一随机空间，接下来将另一随机空间的随机数与剩下的三块进行异或运算得到16比特的Trie树索引值。最后通过查找获得分类规则索引集合，经过线性查找即可确定最终分类规则。下面以表2数据为例说明源和目的IP对分类算法。设由$h[2]$产生的随机数为g，则有$R_1=h[1]\oplus h[3]\oplus h[4]\oplus g$，同理$R_2, R_3, \cdots, R_6$依次产生。

表2 源/目的IP前缀对集合示例

异或运算后的Trie树索引值	目的IP地址		源IP地址	
	$h[1]$	$h[2]$	$h[3]$	$h[4]$
R_1	00101010,10101010,10101010,10101010		00101010,10101010,10101010,10101010	
R_2	00101010,10101010,10101010,10101010		00101010,10101011,10101010,10101010	
R_3	10101010,10101010,10101010,10101010		00101010,10101010,10101011,10111010	
R_4	01101010,10101010,10101010,10101010		00101010,10101010,10101110,10101010	
R_5	00101010,10101010,10101110,10101010		00101010,10101010,10101010,11101110	
R_6	10101010,10101010,10101010,10101010		00101010,10101010,10101010,10101010	

查找分类规则时，首先根据无冲突哈希函数查找表，获得源/目的端口和协议三域所对应的异或前的对集合指针。为了减少查找时间，我们引入异或哈希算法，将64对地址映射为16的关键值，然后首先查找该关键值，确定分类规则索引，最后线性查找每一条规则的源/目的地址，进而完成分类工作。对于源/目的对查找结构如图4所示，树终端用于存放分类规则索引值。

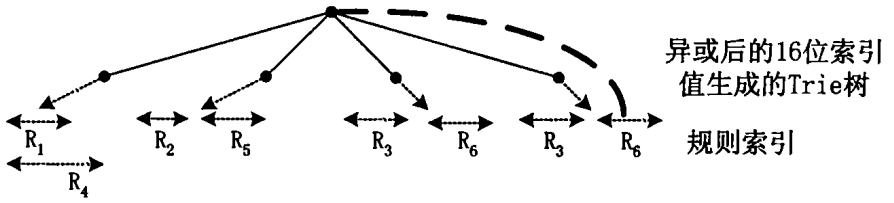

图4 源/目的IP对分类算法

3 评测及结论

我们提出的双哈希的多维查找算法是在无冲突哈希和异或哈希算法的基础上提出的。下面对该算法的平均空间和时间性能做简单分析：

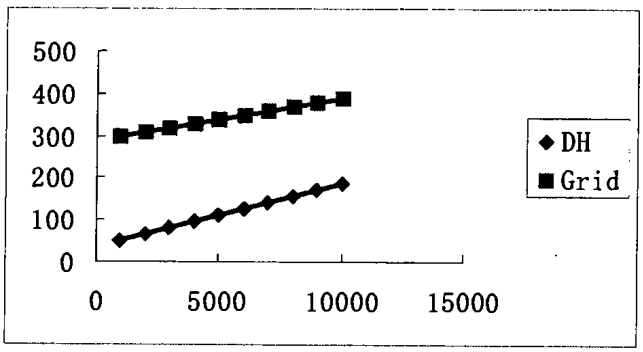

图5 DH和Grid算法时间性能比较图
说明：横坐标表示规则数目，纵坐标表示处理10^5个数据包的分类时间(ms)

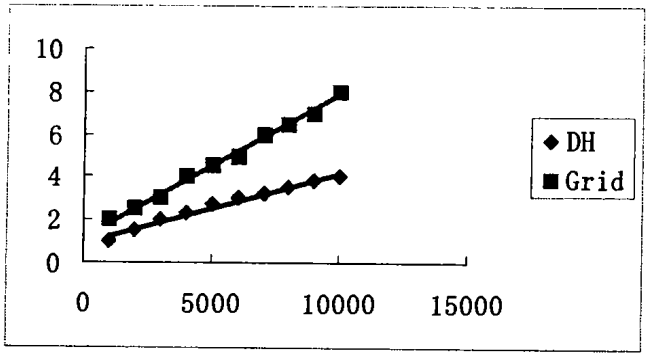

图6 DH和Grid算法空间性能比较图
说明：横坐标表示规则数目，纵坐标表示消耗的最大内存(MB)

对于Grid of Tries算法来说，对于时间性能最坏情况下，确定一条规则需要读存至少4(1+2W)次，查找时间基本上与分类规则数目无关；对于本算法需要4(查找fd表、fs表、fp表和h表最多需读存4次)+1(查找16bit树只需1次)+查找源/目的IP对的次数(仿真时发现最多需30次)。对于空间性能方面，多维Grid of Tries算法需要存储空间为：4个哈希表的大小+2NW，为了保证时间性能，一般4个哈希表要占很大的空间，最坏情况要产生空间爆炸；而对于基于双哈希的多维算法来说所需的存储空间粗略估计为哈希表的大小hash_table(哈希表所占的最大空间，一般情况下规则库中使用的源/目的端口号和协议数目非常少[3])+64kbit(16比特Trie树所占空间)。

我们对于双哈希的多维查找算法在虚拟环境下进行了仿真测试，具体做法是：首先用计算机随机产生规则；接下来按照生成的规则产生数据包头，主要包括目的/源IP、目的/源端口、协议，同时产生的数据包头80%左右能与规则完全匹配；最后用我们设计的算法进行数据包分类，为了消除外界干扰，我们采用求平均时间的办法来计算数据包分类时间，即计算共处理10^5个数据包的分类时间(单位：ms)，来确定最终的时间性能。通过上述算法仿真得到的结果如图5、6所示。其中DH代表本文提出的双哈希算法，Grid表示Grid of Tries算法。

参考文献

1 Gupta P, McKeown N. Packet classification on multiple fields. ACM Computer Review, 1999, 29(4): 146～160

2 Srinivasan V, Varghese G, Suri S, Waldvogel M. Fast Scalable Level Four Switching. ACM Computer Review, 1998, 28(4): 191～205

3 XU Ke, et al. A Non-Collision Hash Trie-Tree Based Fast IP Classification Algorithm. In J. Comput. Sci & Techol., 2002, 17(2): 219～226

4 徐恪，梁志勇，吴建平. 一种基于跳转表的多维IP分类算法. 小型微型计算机系统，2001，22(12)：1409～1413

5 Woo T Y C. A Modular Approach to Packet Classification: Algorithms and Results. In: Gruein R, ed. Proc. of IEEE Infocom2000. San Francisco, CA: IEEE Computer Society Press, 2000. 1210～1217