分布式IP分片处理问题的研究*)

郭方方　杨永田
(哈尔滨工程大学计算机科学与技术学院　哈尔滨 150001)

摘　要　传统的IP分片处理技术只适用于单检查点网络。但随着分布式网络应用的飞速发展,这种传统的TCP/IP协议的基础技术越来越不能适应新的网络环境,而且给网络新技术的推广和应用带来了阻碍。该文在分布式HASH算法的基础上提出了在分布式环境下,多点间协同处理IP分片问题的解决办法,将IP分片赋予某个特定的HASH函数值并由相应的检查点来处理。除此之外还利用折叠异或法提高了HASH算法的计算速度,并且利用前插链表法提高了HASH算法解决冲突问题的效能。通过仿真试验表明该算法可以应用于分布式的网络环境,并且拥有较好的网络适应性和稳定性。

关键词　IP分片,分布式哈希函数,前插式链表,折叠异或

Research of Distributed IP Fragmentation Processing

GUO Fang-Fang　YANG Yong-Tian
(School of Computer Science and Technology, Harbin Engineering University, Harbin 150001)

Abstract　The traditional IP fragmentation processing technology is only suitable for the single checkpoint network. Along with distributed network application rapid development, this traditional TCP/IP protocol foundation technology cannot adapt to the new circumstance. Even it brings the hindrance to the promotion and the application of network new technology. Under the distributed HASH function, a solution is put forward that multi nodes coordinate to deal with IP fragmentation problem in the distributed circumstance. It gives the IP fragmentation a certain HASH function value and processes it by the corresponding checkpoint. In addition, it uses puckering-XOR algorithm to enhance the computing speed of the HASH function. And it used head-inserting linked list algorithm to increase the solving collision efficiency of the HASH function. The simulation experiment indicates that this algorithm can apply in the distributed network environment and has the good network adaptability and stability.

Keywords　IP fragmentation, Distributed HASH function, Head-inserting linked list, Puckering-XOR

1　引言

IP分片的处理是网络节点的基础工作,它直接影响了网络传输的性能、传输的可用性和安全性等方面。尤其对于网络汇聚层的设备,例如边界路由器、防火墙等,分片的处理能力就更为重要。这些需要对分片数据流进行检查处理的中间节点称为检查点。已有多种方法来解决IP分片重组过程的实现、效率及可能的阻碍等问题[1~4]。目前对它的研究还仅限于单个检查点,而分布式网络应用正在成为人类社会越来越重要的组成部分,例如网格计算、多机协同、深度安全检测以及P2P应用等,如何使得分片处理很好地适应分布式网络环境正在成为该问题研究的新的热点。有文献曾提出在各检查点间同步复制会话分片信息,使各点都可对分片进行处理的方法。可为了保持各点信息的一致,要在各点间复制大量数据,因此占用了大量的资源,效率低下[5]。在应用领域,CheckPoint公司的FW-1可对多个检查点的信息进行检查,但其基本原理是将各点得到的信息会聚到服务中心统一处理,因此本质上还是单检查点上的处理活动,依然存在着单失效点和高负载率等固有缺陷[6]。文章针对多点协同工作的分布式网络环境下处理分片的问题进行了研究,提出将接收到的分片计算HASH值,然后根据计算结果将分片分配到负责相应HASH特征值的检查点进行处理的思想。算法将分片的处理分散化,弥补了单检查点的缺陷,发挥了多检查点的分布式处理能力。

2　IP分片处理的基本方法

分片的基本目的是使得通过网络传输的IP包的大小符合传输路径中拥有最小MTU(Maximum Transfer Unit最大传输单元)的中介网络的要求。单点分片算法主要是RFC815及其改进算法[7][8]。分片处理的主要过程是:

1)根据MTU值、IP包长度以及DF位(Don't Fragment不分片标志位)决定是否分片。

2)若要分片则分片的最大长度必须是小于等于MTU且是64的倍数(8个八位组)的最大正整数。然后将每个分片放入一个新的IP包,其包头由原IP包的包头部分复制而来。

3)各新生成的IP分片包独立选路向目的节点传递,最终在目的节点进行重组。如果在限定时间内重组成功则向高层协议提交,否则抛弃分片数据包。

其中需要注意的是目的节点通过IDENTIFICATION以

*)国防科技预先研究项目(413150702)资助。**郭方方**　博士研究生,讲师,研究方向:计算机网络、计算机安全;**杨永田**　教授,博士生导师,研究方向:计算机网络。

及 SOURCE ADDRESS(源站 IP 地址)字段来唯一识别分片属于哪一个 IP 包。

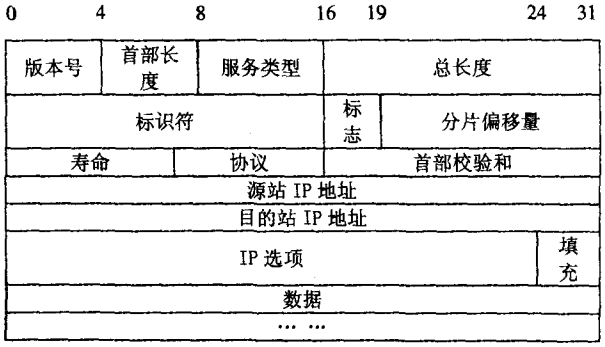

图 1 IP 数据报格式

3 分布式 IP 分片处理算法的设计

3.1 分布式 IP 分片处理算法的整体设计

本文主要是利用分布式 HASH 算法在多个检查点上共同处理分片信息。下面为算法说明：

设：

1)$N\{n_1,n_2,\cdots,n_i,\cdots,n_k\}$为某个网络，$n_i(i,k\in Z^+$且$1\leqslant i\leqslant k)$为 N 中的节点，k 为 N 中的节点数目。

2)$M\{m_1,m_2,\cdots,m_j,\cdots,m_p\}$为 N 的检查点集合，即有 $M\subseteq N$。$m_j(j,p\in Z^+$且$1\leqslant j\leqslant p\leqslant k)$为 M 中的检查点，p 为 M 中检查点的数目。

3)$h(m_j,P_j)$为在检查点 m_j 上执行的 HASH 算法。P_j 是在 m_j 上执行的 HASH 算法的参数组。需要说明的是：各点执行的算法是相同的，这样才能保证计算出来的 HASH 函数值是在同一个值域 A 上得到。

初始化：

1)将 HASH 函数值域 A 依据检查点数目 p 进行分割，分割方法是按 p 等分 A，得到值域片断集合 $S_A=\{A_1,A_2,\cdots,A_j,\cdots,A_p\}$。

2)检查点 m_j 与相应编号的值域片断 A_j 构成一一映射，最终得到一张映射关系表，即序偶对集合 $S_{(M,A)}=\{\langle m_1,A_1\rangle,\langle m_2,A_2\rangle,\cdots,\langle m_j,A_j\rangle,\cdots,\langle m_p,A_p\rangle\}$。

3)将映射关系表 $S_{(M,A)}$ 复制到每一个检查点上。

算法步骤：

当某一个 IP 包的分片 F 到达某个检查点 m_j 时：

1)m_j 提取 F 中相应的信息构成 P_j。

2)m_j 计算 HASH 函数值 $h(m_j,P_j)$。

3)查询映射关系表 $S_{(M,A)}$，查找 $h(m_j,P_j)$落在哪一个值域片断中，与该值域片断构成序偶的检查点是哪个，假设得到的结果是 m_q。

4)将 F 转发到 m_q，由 m_q 处理该分片所属的 IP 包。

说明：

1)该算法将多个检查点构成的非结构化网络构造成了一个结构化网络——无序到达的 IP 包分片被 HASH 算法定向到特定的检查点上进行处理，而且 HASH 算法保证了同一个包的分片都会定向到同一个检查点上进行处理，实现了分布式环境下分片的处理。

2)分片处理过程中要消耗大量的资源，该算法将任务分配给多个检查点各自独立执行，减轻了单个节点的工作负荷，而且更好地利用各个节点的资源。

3.2 HASH 算法的设计

本节将描述如何构造 HASH 函数的参数组，如何计算 HASH 函数，冲突以及分片存储处理。

3.2.1 构造 HASH 函数参数组

HASH 函数的参数组主要用来区分不同的 IP 包分片。由于检查点位于网络路径中间，因此检查点主机协议站要通过源站 IP 地址、目的站 IP 地址和标识符 3 个字段来唯一确定一个 IP 包分片。因此提供给检查点 m_j 上的 HASH 函数的参数组 P_j 为〈SOURCE ADDRESS, DESTINATION ADDRESS, IDENTIFICATION〉。

3.2.2 HASH 函数的计算

在此该算法采用折叠异或的方法计算 HASH 函数的值。

1)将图 1 所示源站 IP 地址字段以及目的站 IP 地址字段从左至右各分为等长的两个部分，得到：

$SA_L=\{SA_0:SA_{15}\}$, $SA_R=\{SA_{16}:SA_{31}\}$

$DA_L=\{DA_0:DA_{15}\}$, $DA_R=\{DA_{16}:DA_{31}\}$

2)然后这 4 个部分与标识符字段进行异或运算，从而得到所求的 HASH 函数值：

$$h(m_j,P_j)=SA_L\oplus SA_R\oplus DA_L\oplus DA_R\oplus \text{IDENTIFICATION}$$

说明：为了增加 HASH 函数值与其自变量之间的联系，即增加一个 HASH 函数值和与之相对应的某个 IP 包分片的相关度，应该让 HASH 函数值的每一位与尽量多的自变量位值相关联，即 HASH 函数值的每一位都是尽量多的自变量位值运算得到的结果。较好的方法是循环迭代法，即参与运算的每一个变量的每一位都要和其它变量的每一位进行异或运算。但是这种方法也将消耗更多的计算资源。因此，该文采用了相对简单的折叠法来进行计算以求获得更快的响应速度——HASH 函数值的每一位只与相对应的 2 位源站 IP 地址位、2 位目的站 IP 地址位和 1 位标识符位有关。图 2 给出了采用以上分割办法的循环迭代法和折叠异或法在总的网络流量一定时，不同分片包比率下响应延迟的比较。可以看出折叠异或法拥有比较好的响应速度。

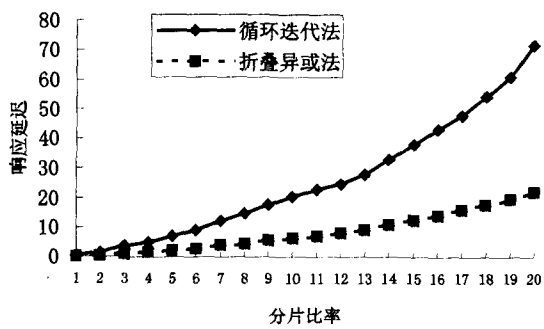

图 2 循环迭代法与折叠异或法响应延迟比较图

3.2.3 分片存储和冲突处理

为了更好地存储分片，每个检查点上利用链表数组 $ChainArray[0..t]$($t+1$ 为值域片断 A 中 HASH 函数值的个数)对分片信息进行存储。数组元素的格式如下：

Keyword	*Head Point*	*Next Point*

说明：

Keyword：该检查点对应的 HASH 函数值域片断值。每一个

值对应着一个数组元素。

Head Point：凡是经过计算得到对应的 HASH 函数值且 SA、DA 和 IDENTIFICATION 三个字段值相同的分片，都依据 FRAGMENT OFFSET 字段的顺序加入到一个表示该 IP 包的分片链表中。*Head Point* 为始终指向该链表的表头节点的指针，初始值为 NULL。

Next Point：指向冲突链表元素的指针。

Next Point 指向的冲突链表元素结构如下：

CollisionHead	*CollisionNext*

说明：

CollisionHead：该指针初始值为 NULL。它指向冲突的一个 IP 包分片链表表头节点。

CollisionNext：该指针初始值为 NULL。它是冲突链指针，在多个产生冲突的链表之间进行链接。

只要采用 HASH 算法，那么冲突就是避免不了的。在该文所述领域，冲突的含义就是经过计算两个不同的 IP 包分片得到了同一个 HASH 函数值。该文采用开散列法进行冲突处理，即链表法。同时由于 IP 流的本地性特征，即同属于一个会话的 IP 分组序列在短时间内相继到达[9]，所以同一个 IP 包的分片也会在短时间之内相继到达。因此对于冲突的 IP 包分片采用前插式链表法。

设：

m_j：为网络中某个检查点

A_j：为 m_j 所属的 HASH 函数值域片断

F_x：为到达 m_j 的一个 IP 包分片信息包

h_x：为 F_x 计算所得的 HASH 函数值

算法描述：

```
Count=0;
While (Count<=LENGTH(mj. ChainArray))
{ if (mj. ChainArray[Count]. Keyword==hx)
    if (mj. ChainArray[Count]. Head Point==NULL)
      { 生成一个新的链表 TableF,将 Fx 放入其中;
        将 mj. ChainArray[Count]. Head Point 指向 TableF 的表头
        节点;
      }
    else
    {if (Fx 是已经存在的链表的一部分)
        将 Fx 插入该链表;
     else
      { 生成一个新的链表 TableF,将 Fx 放入其中;
        将 mj. ChainArray[Count]. Head Point 原先指向的链表插
        入到 mj. ChainArray[Count]. Next Point 指向的冲突链表
        队列的首部;
        将 mj. ChainArray[Count]. Head Point 指向 TableF 的表头
        节点;
      }
    }
  Count=Count+1;
}
```

说明：

由于 IP 流本地特性的存在，靠近链表数组头部的节点很可能很快被再次访问。所以链表数组条目头部的 IP 包分片链表有很高访问频率。因此通过 HASH 函数一次定位后，后续的查找操作就会大大减少，从而加快了访问速度。考察链表数组头部元素，图 3 显示前插式链表法和传统的算法(即冲突的分片链表直接加入到冲突链表尾部)的性能对比。可以看出总流量一定时，不同的分片包比率下前插式链表法确实可以提高分布式 IP 分片算法的性能。

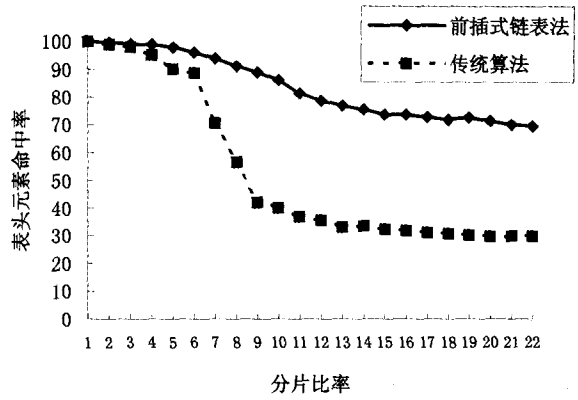

图 3　前插式链表法与传统算法表头命中率比较图

4　试验

仿真试验采用理想化群域模型——网络中有多个自治域，每个域有多个网络节点，域的节点数目相同，其中有一个节点作为检查点，而其它节点将其作为唯一的网络服务接入点，检查点之间采用强连通网络。这种设定屏蔽了网络流量的不均衡问题，也不涉及到复杂的路由问题。通过改变域的数目考察算法的可扩展性，即能否较好地适应网络规模的变化。此外，使用网络近似满载情况下仿真分片流量依给定比例混合注入的方法监测分片信息的处理。而且试验通过随机算法保证对于任一个点的访问是随机的，即所有节点的访问概率是近似相同的。这样使得访问密度不均匀引起的测试结果的不准确性大大降低。

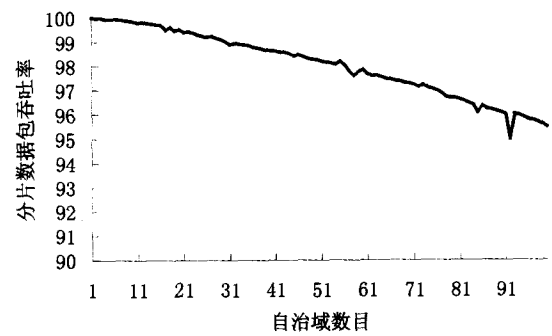

图 4　分片访问成功率分析图

本文设定了一种比较“苛刻”的情况：分片的 IP 包占总流量的 20%且检查点处理分片的超时设定在 30s。在模拟的百兆网络中，域的节点数目在 50 左右，域的个数从 1 到 100 之间进行变化。在试验提供的比较理想的条件下，从图 4 可以看出仿真的分片最终几乎都可以成功到达目的地，吞吐率都在 95%以上。可以说，算法具有较好的网络适应性。同时吞吐率曲线变化幅度比较平缓，并非很剧烈，因此说算法还是比较稳定的。

结论　本文在分布式 HASH 算法的基础上将 HASH 函数的值域映射到所有的检查点上，每个检查点都对应着一部分值域片断。当一个分片到达某个检查点时，该点通过计算出的分片 HASH 函数值，将分片转发到负责相应 HASH 函数值域的检查点上进行处理。具有相同的源、目的地址和标识符的分片都会会聚到同一个检查点上。解决了分布式环境下 IP 分片的处理问题。算法还利用折叠异或法提高了 HASH 算法的计算速度，利用前插链表法提高了 HASH 算

法解决冲突问题的效能。最后通过仿真试验证明该算法是有效的、可行的，可以适应分布式的网络环境。下一步的工作是解决如何在真实的应用环境下使用该算法，譬如在多点安全检测技术里，算法如何与 NAT 相结合等等。

参 考 文 献

1 Lahey K. RFC2923: TCP Problems with Path MTU Discovery [EB/OL]. dotRocket, Inc, 2003

2 林绍太，张会汀，郑力明. IP 分片重组算法(RFC815)的实现及其改进[J]. 计算机工程与设计，2005，26(4)：911～913

3 Luckie M, Cho K, Owens B. Inferring and Debugging Path MTU Discovery Failures [C]. Internet Measurement Conference 2005, Berkeley, CA, 2005

4 Medina A, Allman M, Floyd S. Measuring the Evolution of Transport Protocols in the internet [J]. ACM Computer Communications Review, 2005, 35 (2): 37～52

5 Pearce J P, Maheswaran R T, Tambe M. Local Algorithms for Distributed Constraint Optimization in Dynamic, Anytime Environments [C]. Autonomous Agents and Multiagent Systems 2005, Utrecht, the Netherlands, 2005

6 刘昌华，左爱群. Check Point FireWall - 1 防火墙的特性及应用[J]. 武汉工业学院学报，2004，23(2)：10～13

7 Decwrl J M, Deering S. RFC1191 : Path MTU Discovery [EB/OL]. Stanford University, 1990

8 McCann J, Deering S, Mogul J. RFC1981 : Path MTU Discovery for IP version 6 [EB/OL]. Digital Equipment Corporation, 1996

9 郑卫斌，段中兴，高磊，等. 基于 IP 流本地性的状态检测性能优化方法[J]. 西安交通大学学报，2004，38(4)：413～416

(上接第 17 页)

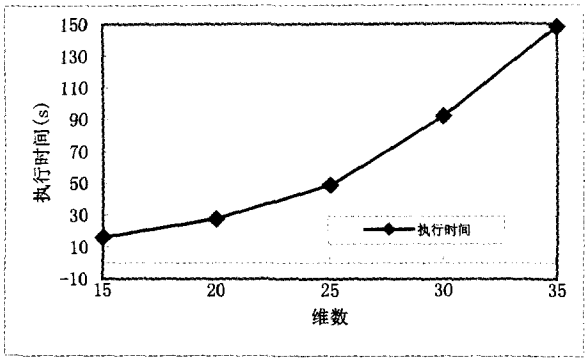

图 6 对数据流维数的伸缩性

为测试算法对数据流时间跨度的伸缩性，生成 B500kC4D30S6 的数据集，如图 5 所示，算法执行时间与数据集大小成线性变化。为测试算法对数据集维数的伸缩性，生成 B500kC4S6 维数为 15、20、25、30 和 35 维仿真数据集。如图 6 所示，算法对数据维数具有较好的伸缩性。实验中 $l=5$，$\xi=15$。

结论 本文研究了 Turnstile 型数据流上的聚类问题，提出高维数据流聚类算法 HT-Stream。该算法采用子空间聚类思想，将数据空间进行网格划分，在一定内存、时间限制下动态估计网格单元统计信息并以倾斜时间窗口进行存储，在用户提出查询时，离线输出聚类结果。HT-Stream 能够提供出现假正的概率保证进而提高聚类的精度，在着重解决数据流高维问题的同时，实现对任意形状分布数据的聚类。实验结果表明，本文提出的算法是可行有效的，进一步提高了算法的实现效率和精度及对高速数据的适应性，并将其扩展到分布式环境是下一步的研究内容。

参 考 文 献

1 Babcock B, Babu S, Datar M, et al. Models and Issues in Data Stream Systems. In: Proceedings of the 21st ACM Symposium on Principles of Database Systems, 2002. 1～16

2 Muthukrishnan S. Data streams algorithms and applications. In: Proc of the 14th Annual ACM-SIAM Symposium on Discrete Algorithms. Philadelphia: Society for Industrial and Applied Mathematics, 2003. 413～413

3 金澈清，钱卫宁，周傲英. 流数据分析与管理综述. 软件学报，2004，15(8)：1172～1181

4 Keogh E, Lin J, Truppel W. Clustering of Time Series Subsequences is Meaningless: Implications for Previous and Future Research. In: Proceedings of the IEE International Conference on Data Mining. IEEE Computer Society Press, 2003. 115～122

5 Bradley P S, Fayyad U M. Refining Initial Points for K-Means Clustering. In: Proceedings of 15th International Conference on Machine Learning. Morgan Kaufmann, 1998. 91～98

6 Vlachos M, Lin J, Keogh E, et al. A Wavelet-Based Anytime Algorithm for K-Means Clustering of Time Series. Workshop on Clustering High Dimensionality Data and Its Applications, at the 3 SIAM International Conference On Data Mining. San Francisco, CA, 2003

7 Rodrigues P, Gama J, Pedroso J P. Hierarchical Time-Series Clustering for Data Streams

8 Guha S, Mishra N, Motwani R, et al. Clustering Data Streams. In: Proceedings of the 41st Annual Symposium on Foundations of Computer Science. Washington, DC: IEEE Computer Society, 2000. 359～366

9 O'Callaghan L, Mishra N, Meyerson A, et al. Motwani. Streaming-Data Algorithms for High-Quality Clustering. In: Proceedings of the 18th International Conference on Data Engineering. Washington, DC: IEEE Computer Society, 2002. 685～704

10 Aggarwal C, Han J, Wang J, et al. A Framework for Clustering Evolving Data Streams. In: Proceedings of the 29th International Conference on Very Large Data Bases. San Francisco: Morgan Kaufmann Publishers Inc, 2003. 81～92

11 孙焕良，赵法信，鲍玉斌，等. CD-Stream——一种基于空间划分的流数据密度聚类算法. 计算机研究与发展，2004，41(Suppl)：289～294

12 Aggarwal C, Han J, Wang J, et al. A Framework for Projected Clustering of High Dimensional Data Streams. In: Proceedings of the 30th International Conference on Very Large Data Bases. San Francisco: Morgan Kaufmann Publishers Inc, 2004. 852～863

13 Agrawal R, Gehrke J, Gunopulos D, et al. Automatic Subspace Clustering of High Dimensional Data for Data Mining Application. In: Proceedings of the 1994 ACM SIGMOD International Conference on Management of Data. New York: ACM Press, 1994. 94～105

14 Bloom B. Space/Time Trade-offs in Hash Coding with Allowable Errors. Communications of the ACM, 1970, 13(7): 422～426

15 Cohen S, Matias Y. Spectral Bloom Filters. In: Proceedings of the 2003 ACM SIGMOD International Conference on Management of Data. San Diego: ACM Press, 2003. 241～252

16 Jin R, Agrawal G. An Algorithm for In-Core Frequent Itemset Mining on Streaming Data. In: Proceedings of the 5th IEEE International Conference on Data Mining. IEEE Computer Society, 2005. 210～217

17 Giannella C, Han J, Pei J, et al. Mining Frequent Patterns in Data Streams at Multiple Time. Next Generation Data Mining, 2003. 97～106