DOI：10.7652/xjtuxb201304005

一种网络冗余流量消除算法

脱立恒[1,2]，倪宏[2]，刘学[2]
（1.中国科学院大学，100049，北京；2.中国科学院声学研究所国家网络新媒体工程技术研究中心，100190，北京）

摘要：针对大量数据片段冗余传输造成网络带宽浪费严重的问题，提出了一种基于动态查找表的冗余流量消除(DYNATABLE)算法。该算法动态统计流量中以不同字节值开头的数据块的冗余率，在保证目标块抽样率的情况下，选取冗余率高的数据块的首字节值为标识，实时更新查找表，根据查找表中的标识从数据包中选出数据块，对已经传输过的冗余数据块进行简单编码，用编码数据替换原冗余数据片段，再对消除冗余流量的数据包进行传输。对比基于最大值选择和基于静态查找表选择等冗余流量消除算法，DYNATABLE算法能跟踪网络数据的动态变化，带来更高的字节节省，平均字节节省率提高到21.8%。

关键词：网络；冗余流量；消除；抽样
中图分类号：TN914.42 **文献标志码**：A **文章编号**：0253-987X(2013)04-0022-06

An Algorithm for Redundant Traffic Elimination

TUO Liheng[1,2], NI Hong[2], LIU Xue[2]
(1. University of Chinese Academy of Sciences, Beijing 100049, China; 2. National Network New Media Engineering Research Center, Institute of Acoustics, Chinese Academy of Sciences, Beijing 100190, China)

Abstract: A dynamic lookup table algorithm for redundant traffic elimination, DYNATABLE in short, is proposed to avoid serious waste of network bandwidth from redundant traffic. DYNATABLE dynamically counts repeat times of data chunks begun with different byte values and uses them as redundancy rates of different byte vales. DYNATABLE selects the values of bytes with higher redundancy rate as markers, and update them in the lookup table. Then chunk selection algorithms select data chunks according to the markers in the lookup table. DYNATABLE encodes the redundant blocks in short form, and then transfers the data from which redundant blocks have been eliminated. Comparisons with the MAXP and the SAMPLEBYTE show that the DYNATABLE gets a high byte saving, and improves the average byte saving rate to 21.8%.

Keywords: network; redundancy traffic; elimination; sampling

随着网络流量的快速增长，流行度高的内容在网络中冗余传输已成为网络内容传输的重要特点。许多技术被用来减少冗余数据的传输，如浏览器缓存和代理缓存等[1]，这些面向对象级(object-level)的技术在很大程度上减少了冗余数据在网络中的传输。Spring等人通过分析网络数据包中的冗余数据片段得出，使用缓存等技术后，仍然检测到39%的冗余数据片段[2]。Anand等人通过对比得出，网

收稿日期：2012-07-18。 作者简介：脱立恒(1984—)，男，博士生；倪宏(通信作者)，男，教授，博士生导师。 基金项目：国家科技支撑计划资助项目(2011BAH11B04)；国家高技术研究发展计划资助项目(2011AA01A102)；中国科学院战略性先导科技专项子课题(XDA06010302)。
网络出版时间：2012-12-27 网络出版地址：http://www.cnki.net/kcms/detail/61.1069.T.20121227.1733.003.html

络数据包级(packet-level)的冗余流量消除(Redundant Traffic Elimination, RTE)比面向对象级的压缩和缓存更高效[3]。近些年，基于数据包的 RTE 已成为新的研究方向，RTE 中冗余数据片段(简称数据块)的选择算法制约着冗余数据的检测效率，如何从网络数据包中选出冗余率高的数据块是 RTE 的研究热点[2-6]。Spring 等人提出基于模运算的块选择算法(MODP)[2]，可以在一定程度上检测出冗余数据块，但该方法可能导致选取的数据块过疏或过密分布。Anand 等人提出了基于最大值选择的块选择算法(MAXP)[3]，该算法倾向选择首字节最大值的数据段，无法针对冗余率高的数据块进行选取。Agarwal 等人提出基于静态查找表的块选择算法(SAMPLEBYTE)[4]，该算法针对特定类型数据包，有较高的冗余数据块抽样率，但无法跟踪网络数据的实时变化。为此，本文提出一种基于动态查找表的冗余流量消除(DYNATABLE)算法，该算法采用贪心算法，动态选取流量中冗余率高的数据块，将选中数据块的首字节作为标识，根据选中标识从数据包中选出数据块。本文算法既可以跟踪网络流量的动态变化，又不会导致选取数据块的过疏或过密分布，具有较高的字节节省率。

1　冗余流量消除

RTE 通常包括以下 4 个步骤。

(1)冗余数据探测。加速器使用块选择算法选出数据包中的数据块，通过指纹算法计算每个数据块的指纹，然后进行指纹匹配。

(2)指纹匹配。将计算出的指纹与加速器缓存中已经存储的历史指纹进行比对，比对成功，对数据片段编码；未比对成功，将指纹、数据块或数据包存入加速器缓存中，供下次指纹匹配使用。

(3)编码。使用短的元数据编码冗余数据块，如$<O,P>$，其中 O 表示冗余块在缓存中的偏移位置，P 表示该数据块在数据包中的起始位置。

(4)解码。接收端加速器根据发送端插入的元数据，对元数据进行解码。

在 RTE 算法过程中，块选择算法决定冗余数据块的选取效率。以下对比不同块选择算法的特点：FIXED 算法每隔 p(p 为目标抽样周期)个字节从数据包中选取一个数据块，由于无法调整数据块选取的起点，该算法对内容有微小变化的数据包具有低的冗余块抽样率，因此通常不被使用；MODP 算法使用哈希函数(rabinhash)计算 Rabin 指纹(Rabin Fingerprint，RF)，并计算连续 w 字节长数据块的指纹 f，选择 $f=0 \bmod p$ 的数据块作为检测片段，该算法可能导致选出的数据块过疏或过密分布，如字节连续相同的数据片段；MAXP 算法以每 p 个字节中字节值最大的字节为起点，选择 w 字节长的数据块作为检验片段，该算法具有稳定抽样率，但对于冗余率高而首字节不是最大值的数据块，该算法无法获得高的检测率。

SAMPLEBYTE 算法使用一个 256 项查找表，256 项索引分别与字节的 256 个值对应，每一项的查找结果 $r\in\{0,1\}$，1 代表该索引对应的字节值被选中为标识，0 代表该索引对应的字节值不被选中，查找表中固定 k 个字节值被选为标识。SAMPLEBYTE 算法在分析数据包时，逐个字节扫描，如果扫描到的字节对应字节值的查找结果为 1，则以该字节所在位置为起点，选出数据块，该字节称为该块的标识。计算选中数据块的指纹，按照 RTE 的处理步骤继续处理。一旦选中某个数据块，则以标识所在位置为起点，跳过 $p/2$ 个字节重新开始字节扫描。

SAMPLEBYTE 算法使用离线算法统计训练数据，得到固定 $k=8$ 个标识，分别为 0、32、48、101、105、115、116、255。

文献[2]指出，在 $p=32$ 时，系统可获得吞吐量和字节节省间更好的平衡，之后的研究[7-8]多以文献[2]的抽样率作为理论抽样目标。如果网络数据包中所有字节值均匀分布，则 SAMPLEBYTE 算法对数据块的抽样率为 8/256；如果网络数据包中字节值非均匀分布，将导致 SAMPLEBYTE 算法过抽样或低抽样。

2　基于动态查找表的网络冗余流量消除算法

2.1　动态查找表更新算法

由以上分析得出，抽样率与标识个数以及标识在数据包中出现的概率有关，RTE 效率与选择哪些标识有关，为了动态跟踪抽样率，同时选择合适的标识，需要实时跟踪网络数据的变化情况。为了获得精确的抽样率，实时统计网络数据包中不同字节值的出现概率，以便调整查找表 L_{lpt} 中标识的个数；为了获得高的冗余数据检测效率，实时统计以不同字节值为标识的数据段的冗余频率，以数据块冗余出现频率高的字节值为基础，预测下阶段哪些标识对于提取冗余数据块更加有效。

假设统计的 256 个字节值的出现概率分布为 $[p_0, p_1, p_2, \cdots, p_{255}]$，以 256 个不同字节值开头的数据段的冗余概率为 $[f_0, f_1, f_2, \cdots, f_{255}]$，选择不同标识的过程即为选择 $x_0, x_1, x_2, \cdots, x_{255}$，使得式(1)、式(2)成立

$$\sum_{0}^{255} p_i x_i \leqslant \frac{1}{p}, \quad x_i \in \{0,1\} \tag{1}$$

$$\max \sum_{0}^{255} f_i x_i \tag{2}$$

式(1)表示：选中的所有标识的抽样率小于等于 $1/p$；式(2)表示：在式(1)限制条件下，选择使得不同字节值开头的数据段的冗余概率和最大的字节值作为查找表的标识。使用冗余概率和最大的标识所选择出的数据块的冗余率越高，RTE 效率就越高。选择标识的问题属于背包问题，本文使用贪心(greedy)算法求解该问题。按照不同字节值的冗余概率降序排列，选择使得总的抽样率接近 $1/p$ 的字节值作为标识。

2.2 以不同字节值开头的数据块的修正冗余频率计算方法

DYNATABLE 算法统计不同字节值在数据包中的出现概率，方法是：在使用 RF 算法计算指纹的过程中，实时记录不同字节值 A 出现的次数 M_A，用该次数除以该段时间数据总长度，即为出现概率。要统计以不同字节值开头的数据块的冗余频率，比较困难，如果按每字节逐个统计所有数据块，可以获得以不同字节值开头的数据块的冗余频率。但是，在进行 RTE 时，每次选中数据块后，需要跳过 S 长的数据片段，以防止数据块的过抽样，此时，按每字节统计的数据块冗余频率将不再准确，所以需要对按每字节统计的冗余数据块的冗余频率统计方法进行修正。假设在按字节统计的过程中，以字节 A 开头的前后 2 个冗余数据块的距离为 L，后一个数据块能否被选出，与距离前一个以 A 开头的数据块的距离有关，前后 2 个数据块的距离 $L<S$ 时，如果前一个数据块被选中，在跳过 S 的过程中，后一个数据块将被跳过，该重复数据块处理流程如图 1 所示。

实际选择中，前一个数据块被选中也存在一定概率，其与更前面的数据块的选择有关，但针对后一个数据块，其距前一个数据块的距离 L 越大，意味着离更前面的数据块的距离越大，被选中的概率越大；前后 2 个以 A 开头的数据块距离 $L \geqslant S$ 时，后一数据块肯定能被选中，即后一个数据块对以 A 开头的数据块冗余频率贡献(contribute)为

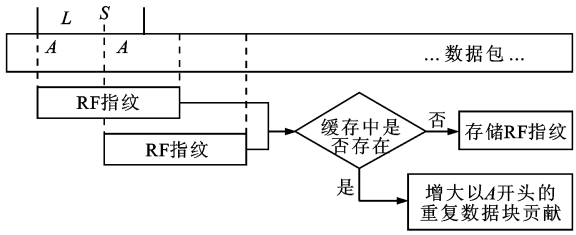


图 1 重复数据块处理流程

$$c_A(L) = \begin{cases} L/S, & L < S \\ 1, & L \geqslant S \end{cases} \tag{3}$$

累加以 A 开头的数据块冗余频率贡献得到以 A 开头的数据块修正冗余频率 C_A。

利用布隆滤波器(Bloom Filter，BF)[9]能快速判断某个元素是否属于某个集合的特点，在统计以不同字节值 A 开头的数据块的修正冗余频率时，将指纹放入布隆滤波器实现的集合 G_A 中。

2.3 DYNATABLE 算法流程

综上，DYNATABLE 算法如下。

输入参数：t_{pkt} 为数据包指针，w 为数据段长度，N 为数据总长度，且 $N>w$，i 为临时变量

```
(1) for(i=0; i< w-1; i++)
(2)   f=rabinhash (t_pkt, i, f);
(3)   ++M_A;
(4) for(i=w-1; i < N; i++)
(5)   A=t_pkt[i];
(6)   f=rabinhash (t_pkt, i, f);
      /* rabinhash 计算以 i-w 开头的 w 字节长的数据段的 RF 指纹 f */
(7)   ++M_A;
(8)   if(G_A contains f)
(9)     C_A+=contribute (t_pkt[i-w]);
(10)  else
(11)    store f in G_A;
(12)  if(every fixed data length)
(13)    L_lpt=greedy(M_1-255, C_1-255, p);
(14)    update L_lpt to shared memory;
```

2.4 布隆滤波器设计

BF 的参数设计对统计效率至关重要。BF 由一个长度为 m 的二进制向量和 q 个随机映射函数组成，它将输入元素映射到二进制向量的 q 个比特位上，并将比特位置 1，检测某输入元素是否已经存在，首先计算该元素的映射，然后将该映射与二进制向量进行比对，如果 q 个映射比特位全部置 1，说明元素已经存在，否则元素不存在。BF 可以保证所有

已经插入过的元素肯定能被检测出，但存在一定的误识别率 p(False Positive Rate，FPR)。设存入元素数为 n，则 p 与 m、q、n 的关系为

$$p \approx (1-\mathrm{e}^{-qn/m})^q \tag{4}$$

为保证计算的高效性，选 $q=2$，BF 存储的最大元素数 $n=10^6$。为使 FPR 控制在 $p\leqslant 1\%$，由式(4)得出，二进制向量长度 m 应取 2.5 MB。

2.5　256-LSA 缓存替换算法

SAMPLEBYTE 算法中使用 FIFO(First In First Out)的缓存替换算法，Halepovic 等人提出使用 LSA(Least Savings with Aging)缓存替换策略[7]。由于 DYNABYTE 算法实时动态更新查找表，当查找表中标识发生变化时，已被替换出的标识对应的数据块将不可能再被命中，LSA 算法无法迅速替换出该部分数据块。DYNABYTE 算法将缓存拆分成 256 份，每份缓存对应一个字节值，不同数据块存入其标识对应的缓存中，当查找表发生变化时，被替换出的标识的缓存将不被再次访问。

3　实验与仿真

本节讨论 DYNATABLE 算法的实验及仿真结果。实验设备为 DELL R710 服务器，配置为 16 核 CPU、4 GB 内存、500 GB 硬盘，操作系统为 CentOS 5.5。DYNATABLE 算法的查找表更新算法为块选择算法提供动态的查找表，但查找表更新算法与块选择算法运行时相互独立，两者分别在 2 个独立进程中并行运行。DYNATABLE 算法的查找表更新进程在更新查找表后，将其更新到共享内存区，同时通知块选择算法进程实时更新查找表。

3.1　实验数据

为了准确对比不同块选择算法的效率，抓取表 1 中 4 种情况的流量作为分析数据。抓取数据总流量大小为 130.6 GB。

表 1　统计数据信息

跟踪数据	数据来源	时长/h	数据流量类型	数据量/GB
A	国家实验室	12	入口	12.6
B	国家实验室	12	出口	9.5
C	多媒体发布	12	出口	93.3
D	web 服务站点	24	出口	15.2

3.2　缓存替换算法

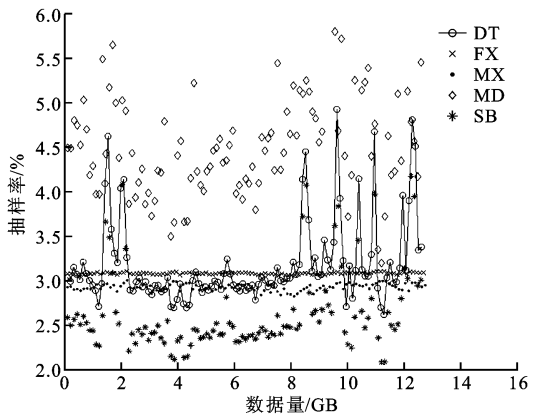
实验中分别使用 LSA 和 256-LSA 这 2 种缓存替换策略，对 4 段跟踪数据进行分析，结果如表 2 所示。由表 2 可见，256-LSA 缓存替换方法比 LSA 替换方法的字节节省率提高约 20%。以下实验中，均使用 256-LSA 缓存替换策略。

表 2　缓存替换性能对比

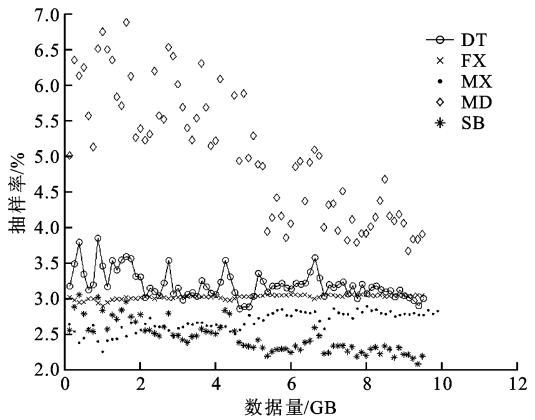
跟踪数据	字节节省率/%		性能提高率/%
	LSA	256-LSA	
A	12.32	15.06	22.24
B	19.14	22.9	19.64
C	9.95	11.86	19.2
D	31.51	38.43	21.96
平均值			20.76

3.3　抽样率

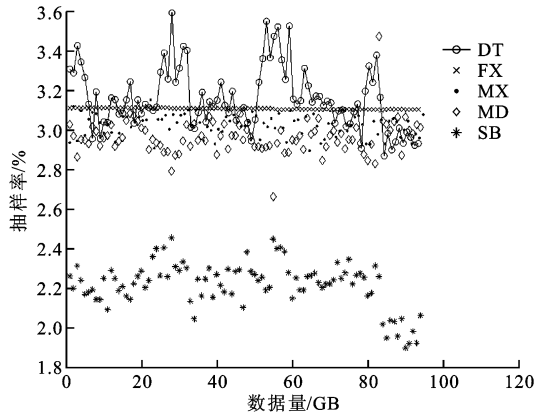
用 DT 代表本文 DYNATABLE 算法，FX 代表 FIXED 算法，MX 代表 MAXP 算法，MD 代表 MODP 算法，SB 代表 SAMPLEBYTE 算法，图 2 显示了 5 种算法的实际抽样率变化情况，其中 FIXED 和 MAXP 的抽样率相对比较稳定，MODP 抽样率过高，SAMPLEBYTE 的抽样率较低。DYNATABLE 算法能够动态调整抽样率，使抽样率在目标抽样率附近波动。



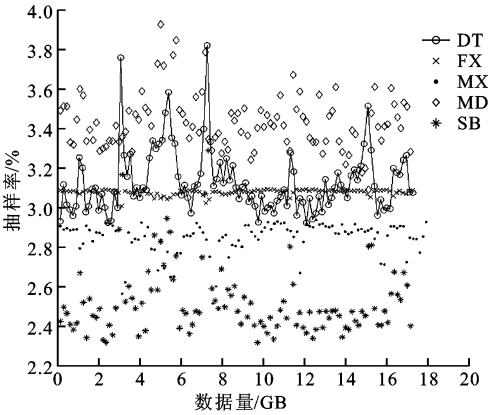
(a)跟踪数据 A



(b)跟踪数据 B



(c)跟踪数据 C



(d)跟踪数据 D

图 2 5 种算法的抽样率对比

3.4 字节节省

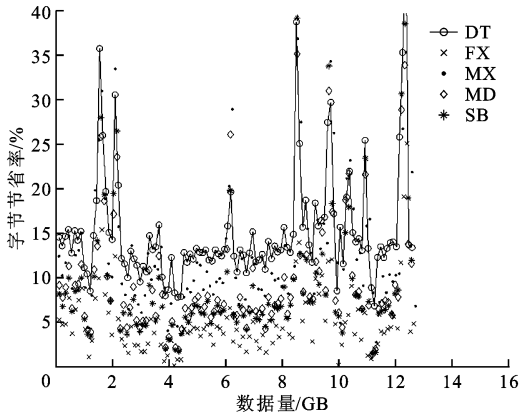
字节节省率定义为冗余消除后传输字节数与冗余消除前的字节数的比值。表 3 和图 3 对比了 5 种算法的字节节省率。可以看出,DYNATABLE 算法带来了更高的字节节省率,对于数据 A、B、D,DYNATABLE 算法比 MAX 算法平均提高 16.1%,对于数据 C,DYNATABLE 算法比 SAMPLEBYTE 算法提高 38.8%。

表 3 5 种算法的字节节省率对比

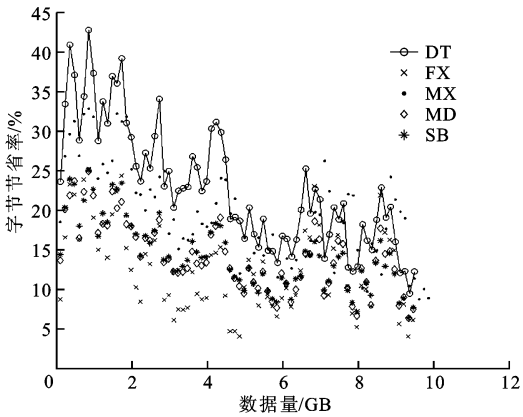
跟踪数据	字节节省率/%				
	MD	MP	FX	SB	DT
A	9.43	12.84	5.04	9.03	15.06
B	14.14	19.25	12.67	14.34	22.9
C	7.29	7.71	5.35	7.82	10.86
D	25.96	34.25	21.68	25.43	38.43
平均值	14.21	18.51	11.19	14.16	21.81

3.5 运行时间

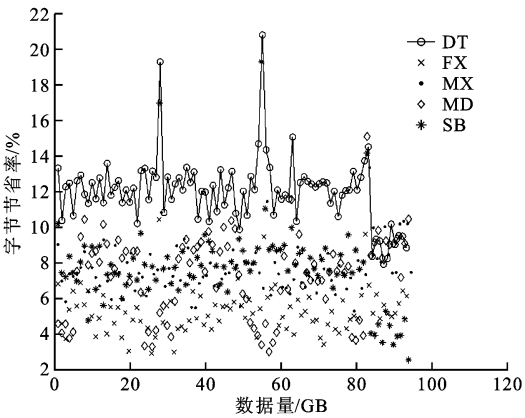
表 4 同时统计了 5 种算法处理每 GB 数据的 CPU 运行时间。DYNATABLE 算法的查找表更新算法和块选择算法分别独立运行,表中 DYNATABLE 算法的 CPU 时间表示取查找表更新算法的



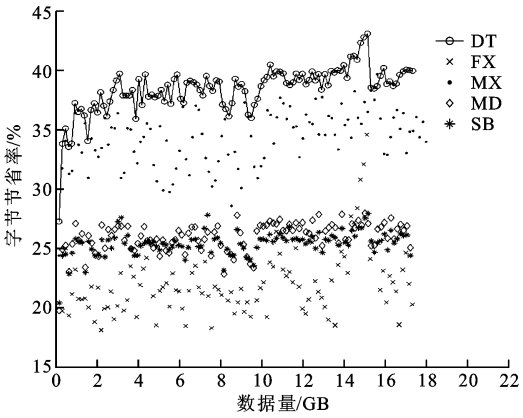
(a)跟踪数据 A



(b)跟踪数据 B



(c)跟踪数据 C



(d)跟踪数据 D

图 3　5 种算法的字节节省率对比

CPU 运行时间和块选择算法的 CPU 运行时间中的较大值。CPU 时间消耗包括计算指纹、缓存查找和插入等操作开销。块选择算法的计算量主要集中在对缓存的查找和插入操作；DYNATABLE 算法的查找表更新算法，计算量集中在数据块的查找或插入 BF 集合，通过实验，DYNATABLE 的查找表更新算法和块选择算法的运行时间基本相当。虽然 DYNATABLE 算法增加了计算量，但并未减小处理数据的吞吐量。

表 4　5 种算法处理数据的 CPU 运行时间

跟踪数据	CPU 运行时间/s				
	MD	MP	FX	SB	DT
A	37.28	23.36	24.56	21.04	25.53
B	41.2	21.36	24.24	19.68	24.9
C	23.76	24.16	24.8	17.84	25.44
D	27.52	22.8	24.64	20.16	25.17
平均值	32.44	22.92	24.56	19.68	25.26

4　结　论

本文提出了一种基于动态查找表的 RTE 算法(DYNATABLE)，该算法实时统计网络数据中以不同标识开头的数据块冗余率，选取冗余率高的标识，动态更新查找表，以选取更多的冗余数据块，在网络数据传输时，可获得高的字节节省；在统计以不同标识开头的数据块冗余率时，使用布隆滤波器存储已出现的数据块的指纹，缩短算法运行时间；DYNATABLE 算法中使用 256-LSA 缓存替换算法更新缓存，以提高缓存命中率。仿真实验表明，DYNATABLE 算法可以在不减少数据处理吞吐量的条件下，有效提高网络数据传输中的字节节省率，从而提高广域网传输效率。

参考文献：

[1] GREVERS T, CHRISTNER J. Application acceleration and wan optimization fundamentals [M]. Indianapolis, IN, USA: Cisco Press, 2007: 170-205.

[2] SPRING N T, WETHERALL D. A protocol-independent technique for eliminating redundant network traffic [J]. ACM SIGCOMM Computer Communication Review, 2000, 30(4): 87-95.

[3] ANAND A, MUTHUKRISHNAN C, AKELLA A, et al. Redundancy in network traffic: findings and implications [C]// Proceedings of the Eleventh International Joint Conference on Measurement and Modeling of Computer Systems. New York, USA: ACM, 2009: 37-48.

[4] AGGARWAL B, AKELLA A, ANAND A, et al. EndRE: an end-system redundancy elimination service for enterprises [C]// Proceedings of the 7th USENIX conference on Networked Systems Design and Implementation. Berkeley, CA, USA: USENIX Association, 2010: 18-28.

[5] RABIN M O. Fingerprinting by random polynomials, technical report: TR-15-81 [R]. Cambridge, MA, USA: Harvard University. Center for Research in Computing Technology, 1981: 18-51.

[6] 敖莉，舒继武，李明强. 重复数据删除技术 [J]. 软件学报，2010，21(5)：916-929.
AO Li, SHU Jiwu, LI Mingqiang. Data deduplication techniques [J]. Journal of Software, 2010, 21(5): 916-929.

[7] HALEPOVIC E, WILLIAMSON C, GHADERI M. DYNABYTE: a dynamic sampling algorithm for redundant content detection [C]// Proceedings of 20th International Conference on Computer Communications and Networks. Piscataway, NJ, USA: IEEE, 2011: 1-8.

[8] ANAND A, SEKAR V, AKELLA A. SmartRE: an architecture for coordinated network-wide redundancy elimination [J]. ACM SIGCOMM Computer Communication Review, 2009, 39(4): 87-98.

[9] BLOOM B H. Space/time trade-offs in hash coding with allowable errors [J]. Communications of the ACM, 1970, 13(7): 422-426.

(编辑　刘杨)