



骨干网冗余数据消除系统的研究与实现^{*}

刘保健¹, 黄 杰¹, 王宝生¹, 张潇晓²

(1. 国防科学技术大学计算机学院 长沙 410073; 2. 西北工业大学计算机学院 西安 710072)

摘 要: 当今的广域网传输过程中存在着大量的重复数据, 这些重复数据使得广域网带宽被严重浪费。针对这一问题, 提出了一种基于应用层协议识别的冗余数据消除策略。通过将数据缓存及应用层协议识别技术相结合, 找出广域网传输过程中的冗余数据, 尽可能使广域网中的相同数据只传输一次, 从而降低冗余数据传输对广域网带宽的过度浪费。

关键词: 计算机网络; 冗余数据消除; 协议识别; 数据缓存; 缓存管理

doi: 10.3969/j.issn.1000-0801.2013.09.002

Research and Achievement of Redundancy Elimination of Backbone Network

Liu Baojian¹, Huang Jie¹, Wang Baosheng¹, Zhang Xiaoxiao²

(1. School of Computer Science, National University of Defense Technology, Changsha 410073, China;

2. School of Computer Science, Northwestern Polytechnic University, Xi'an 710072, China)

Abstract: Nowadays, WAN transmission process has a lot of duplicate data, these duplicate data make bandwidth of WAN to be serious wasted. In order to solve this problem, a redundant data elimination strategy based on application layer protocol identification (PI-RE) was put forward. Using the combination of data caching and the application layer protocol identification technology, the redundant data of WAN during transmission were found out, so that the same data transferred only once in WAN as much as possible, to reduce the redundant data transmission excessive waste of WAN bandwidth.

Key words: computer network, redundancy elimination, protocol identification, data cache, cache management

1 引言

随着互联网中传输数据体积的逐渐增大, 数据传输对网络带宽构成了巨大的挑战, 而网络传输性能的提升, 又不能仅依靠网络带宽的提升。数据研究表明, 网络传输的速率并不是随着带宽的增加而一直增加的。因此, 广域网 (wide area network, WAN) 传输优化技术成为近些年来研究领域的热点问题。

研究发现, 广域网中传输的数据存在着大量重复。在

不同环境中, 重复数据比例也不一样。在普通高校网络中, 存在大约 25% 的冗余数据, 而在企业网络中, 则存在大约 50% 的冗余数据^[1]。导致重复数据产生的原因有很多, 譬如, 大量用户访问相同的 Web 资源, P2P 形式的文件下载与上传, 还有现在大型公司对文档的统一化管理, 每一份文档或者其他数据资源在下载之后修改很小的部分, 会被再次上传到服务器等。因此, 需要寻求一种策略, 降低这些无用的重复传输, 从而节省网络带宽。而大量重复数据的存在, 必将使冗余数据消除成为一种非常行之有效的广域网优化技术。

^{*} 高等学校博士学科点专项科研基金资助项目 (No.20114307110006)

很多系统都研究了如何高效地剔除传输链路中的重复数据,从而提升网络传输效率。一些系统在应用层和对象级做优化,例如 Web 缓存^[2]和 P2P^[3]缓存,它们将频繁访问的对象和请求存储在缓存中。基于字典的算法,譬如 GZIP^[4],从对象中去除重复字节。很多研究者都在积极探索应用层和对象级的有效方法,以改进算法,优化设计^[5-7]。

近年来,一种新的不依赖于协议的冗余数据消除算法被开发和验证,用于消除网络数据分组中的重复字符串,这一理念首次在参考文献[8]中被提出。这一理念已逐步被各广域网加速厂商应用到它们的设备中,也有越来越多的学者致力于对算法的改进,以提升效率。

广域网传输数据流中的冗余数据消除主要面临着以下两个问题。

- 广域网中的数据流是实时流量,必须被及时处理,具有较高的计算效率,不然就会大幅增加数据处理时延。
- 广域网中的传输数据流具有时间的延续性,它会一直存在,所以基于数据流的冗余数据消除并不能像文件备份系统中的冗余数据消除策略,将所有数据都存储在系统中以待扫描检测,所以如何高效地选取存储在内存中的数据也是极其重要的问题之一。

针对以上两个主要问题,提出了基于应用层协议识别的冗余数据消除(protocol identification redundancy elimination, PI-RE)算法。在广域网传输过程中,不同应用层协议下传输的数据有着不同的冗余度,所以基于应用层协议识别的冗余数据消除策略必将提升冗余消除度。

2 相关研究

现在,冗余数据消除技术需要部署额外的硬件或者软件资源,例如 Web 缓存已经部署并且应用了若干年。然而,对象级别的缓存并不能消除所有的冗余数据^[8],传统的对象级别的冗余数据消除技术针对仅对对象做出细微改变的冗余检测并没有很好的效果。因此,要想更多地检测出冗余数据,必须在更小粒度的数据中检测冗余,将数据流划分为连续的字节序列,从而检测冗余。

这样,冗余数据消除技术就被划分为以下几个技术问题:字节序列要取多长;如何唯一地表示字节序列;由于内存有限,什么样的字节序列更应该存储下来;如何判断两个字节序列是否相同。这些问题都是冗余数据消除技术需要考虑的技术细节。

2.1 变长分块算法

变长分块 CDC(content defined chunking)算法,是冗余数据消除技术中通用的数据块划分算法,数据块可以是定长,也可以是变长。定长 CDC 如图 1 所示。

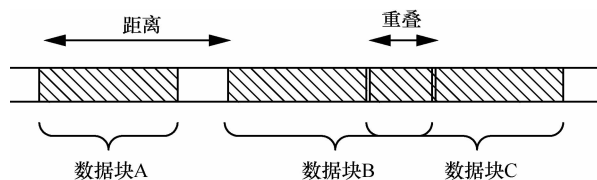


图 1 定长 CDC 算法

在广域网传输数据流中,使用 CDC 联合指纹计算进行冗余消除,CDC 使用定长数据块,指纹计算采用 SHA-1 或者 Rabin 指纹。

2.2 数据块的选取算法

因为内存有限,选择什么样的数据块存入缓存,是整个冗余数据消除算法的核心问题。

- MODP:是在参考文献[8]中提出的,它的选取方法是将每一个数据块的指纹值对 P 取余,选择计算结果等于 0 的数据块。 P 是 2 的指数值,这种算法不受文件微小变化的影响,但是对于数据块的选择不可控。
- WINN:这一算法的数据块选取方式是通过计算每一个数据块的指纹,在每一个固定的长度区间,选择指纹值最大的那一个数据块。和 MODP 相比,这一算法有着固定的采样率。这一选取算法很好地提升了冗余数据的检测,尤其在 HTTP 页面上^[9]。
- MAXP:MODP 和 WINN 的最大问题是需要计算每一个数据块的指纹,从而产生较大的计算量。MAXP^[9]仅在需要的时候才计算指纹,MAXP 虽然基于 WINN 算法,但是它并不是通过比较指纹的局部最大值,而是通过数据块中的数据字节来寻找局部最大值,减少了指纹计算的开销,在冗余度检测方面的贡献,它和 WINN 是一样的。
- 基于内容感知的数据块选择,不同的数据产生冗余的潜在能力是不同的,譬如 HTML 文档和 MP3 文件,HTML 文档大概有 70% 的数据是冗余的,而 MP3 只有不到 1%^[10]。因此,在 MP3 和 HTML 文档的数据块采样是不应该相同的,HTML 文档应该有更高的采样率,而 MP3 应该忽略不计。



3 PI-RE 系统

3.1 PI-RE 的基本流程

为了在以后可以方便地更改算法,采取在应用层执行数据块处理的策略,在每一次的算法升级过程中,并不需要重新编译内核,在内核和应用层通过 Netlink 创建通信链路,将发送端网络层的数据分组原封不动地发往用户层,在用户层进行数据处理。

在发送端的处理流程中,数据块的选取是整个冗余数据消除技术的重点问题,选取算法的好坏,直接影响着冗余数据的检测。

缓存管理也是冗余数据消除技术中另一个不可或缺的重点内容,因为缓存空间的有限性,必须高质量地使用缓存,将最有用的数据存入缓存。

PI-RE 系统实现的发送端基本结构如图 2 所示。

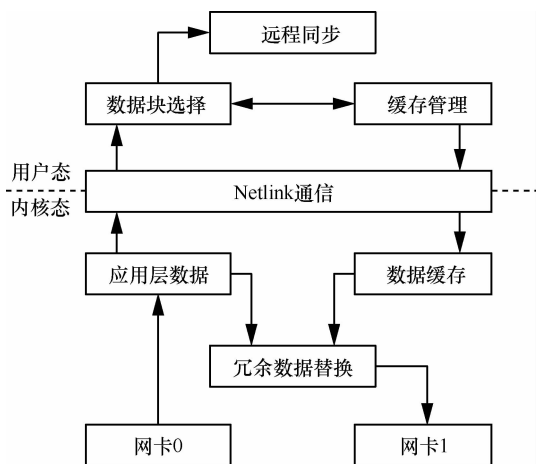


图 2 发送端基本结构

远程同步模块用于与接收端同步数据缓存项。数据块选取和缓存管理的详细描述将在第 4 节给出。

PI-RE 的接收端基本结构如图 3 所示。

远程同步模块用于与发送端同步缓存,网卡 0 接收发送端发送来的数据,结合缓存,通过数据还原模块还原成

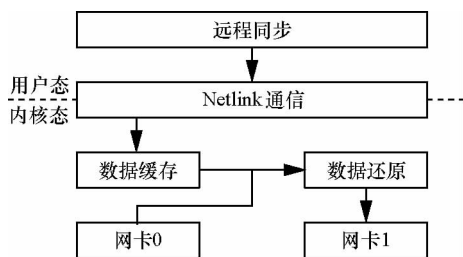


图 3 接收端基本结构

为原始数据再经网卡 1 发送出去。

3.2 协议识别

协议识别的方式主要有基于端口的协议识别和基于报文载荷内容的协议识别,前者在网络发展早期协议识别率高,但目前绝大多数应用程序多采用随机端口,因此在协议识别的准确率上已无法满足需求,而基于报文载荷内容的协议识别的准确率比前者高^[1],因此本系统采用的是基于报文载荷内容的协议识别技术。

系统的协议识别是基于 PCRE 的协议识别,通过读取 IP 分组载荷的内容对 OSI 7 层中的应用层协议进行匹配,其关键是获取标识协议类型的正则表达式。而基于 PCRE 的正则表达式首先对正则表达式进行编译,然后通过匹配引擎来实现。

系统采用 L7 过滤器所提供的特征字符串,匹配引擎采用 PCRE,报文经过 PCRE 匹配,匹配到的报文予以标记相对应的编号,用于生成统计流量信息。

3.3 指纹的生成

指纹的生成考虑的主要问题是计算的开销,指纹计算方式也有很多种,常用的计算方式有 MD5、SHA-1 和 Rabin 指纹^[12],但是 MD5^[13]和 SHA^[14]在计算数据流中的数据块指纹方面计算效率不是很好,因此,选择 Rabin 指纹。

计算一个字节序列的 Rabin 指纹,字节序列为 $t_1, t_2, t_3, \dots, t_\beta$, 计算如下:

$$RF(t_1, t_2, t_3, \dots, t_\beta) = (t_1 p^\beta + t_2 p^{\beta-1} + t_3 p^{\beta-2} + \dots + t_{\beta-1} p + t_\beta) \bmod M \quad (1)$$

P 和 M 是常整形数,这个形式的表达式可以计算每一个长度为 β 的子字符串 $\{t_1, t_2, \dots, t_\beta\}, \{t_2, t_3, \dots, t_{\beta+1}\}, \dots$, 如果连续计算指纹,在计算完成第一个指纹后,接下来计算如下:

$$RF(t_{i+1} \dots t_{\beta+i}) = (RF(t_i \dots t_{\beta+i-1}) - t_i p^\beta) p + t_{\beta+i} \bmod M \quad (2)$$

这样就可以大大提升指纹的计算效率。

3.4 数据块的选取

由于缓存空间的有限性,选择什么样的数据块存入缓存,将是一个至关重要的问题,因为将每一个字节都存入缓存是不现实的。前面第 2 节提到的若干基本算法对所有数据都一视同仁,而由于不同数据对冗余度的潜在贡献并不是相同的,所以后来有学者提出了基于内容感知的数据块选取算法。通过对冗余数据的统计研究发现,重复传输的数据不仅和其内容有关,而且随着应用层协议的不同,对冗余度的贡献也不相同。

既然重复数据与上层的应用层协议相关,就可以把应

用层协议作为数据块选取的限定条件,提升其性能。而且统计表明,不同协议类型的数据,在总传输数据中的比例是不同的。所以,将应用层协议类型作为采样频率的限定条件,必将可以提升性能。不同协议的数据在广域网中所占比例及数据冗余比例见表 1。

表 1 关键协议冗余度贡献

协议类型	传输比例	冗余比例
HTTP	57.31%	48.37%
FTP	1.9%	7.5%
HTTPS	3.59%	4%
SMTP	3.27%	70.63%
DNS	0.83%	47.99%
RTSP	1.34%	24.40%
其他	31.76%	10.22%

首先,每接收到一个应用层数据分组,先分析其应用层协议类型,然后再根据其不同的协议类型调整数据块采样频率,潜在冗余贡献越大,采样周期越短,采样越多。而对于那些几乎对冗余不会有什么贡献的数据,直接略过。

3.5 缓存管理

缓存空间的有限性使得必须有高效的缓存管理手段,才能更加高效地利用有限的缓存空间。

现在通用的缓存管理算法主要包括先进先出(FIFO)、最近最少使用(LRU)。

FIFO 算法实现简单,几乎不存在开销问题,但是对于缓存的管理效果不是很好,而 LRU 较 FIFO 有更好的效果提升,但是代价也有提升。LSA 是参考文献[10]提出的一种缓存置换算法,该算法在基于使用频率的置换算法基础上加入了基于时间的衰退,因为单单基于使用频率的置换算法,容易造成缓存污染。缓存污染是指,如果缓存中的某些项在一段时间内有着较高的活跃度,但是随着时间的递增,这些曾经活跃的缓存项不再有用,但是它们的使用频率仍旧很高,导致无法被置换出缓存。LSA 加入时间因素,该算法设定一个时间值,每当一个固定的时间到达,就清空缓存,因为算法的提出者认为,某一数据块的重复具有时间局限性,LSA 算法可以有效地避免缓存污染。本文将 对缓存管理做出进一步的优化。

LSA 虽然较 LRU 在提升缓存管理的效果上有所进步,但是该算法在处理缓存项的问题上过于激进。

本文提出的改进算法将剔除时间因素,通过添加计数值和缓存清理触发机制来维护缓存。加入时间因素可以有

效地避免缓存污染,通过缓存清理触发机制,同样可以有效地避免缓存污染。维护数据块的重复命中计数可以更加高效地提升缓存的利用效率。

算法基本思路:在每一个数据块加入缓存的同时,为其添加一个关联属性,用于记录该数据块每一次产生的重复传输。当缓存被填满时,该记录值是某一个缓存数据块是否被清理出缓存的有效评价指标。每次当缓存被填满时,触发缓存清理机制,遍历缓存中的缓存项,将缓存中重复命中次数为零的缓存项清理出缓存,并将重复命中次数不为零的缓存项的重复命中次数清零。算法的具体实现步骤在第 4 节给出。

4 PI-RE 的实现

4.1 数据块的选取

广域网上传输的数据流具有其独有的特性,所以在消除冗余数据的过程中,它并不能像分布式系统中的文件备份传输那样,拥有数据副本,数据流的典型特征就是它的无限性。因此,必须选择一种高效的数据缓存策略,数据块的选取就是其中最为重要的步骤之一。

在数据块选取模块中,重要的数据结构主要有两个:一个用来检测重复命中,另一个用来存储缓存数据项。

在该系统中,检测命中使用散列表结构,散列表节点的结构为:

```
struct HashNode{
    unsigned int fingerprint;
    unsigned short key;
};
```

其中 *fingerprint* 为 32 bit 的 Rabin 指纹, *key* 为缓存数据块的唯一标识符,也代表了该指纹对应的数据块在缓存数组中的位置。

数据缓存使用数组来完成,数组大小为 65 535,数组节点的结构为:

```
struct CacheNode{
    unsigned short hit;
    unsigned int fingerprint;
}
```

其中, *hit* 为该缓存数据重复命中的次数, *fingerprint* 为缓存数据所对应的指纹值。

数据块选取模块的具体执行步骤如图 4 所示。
数据块选取的主要步骤如下:

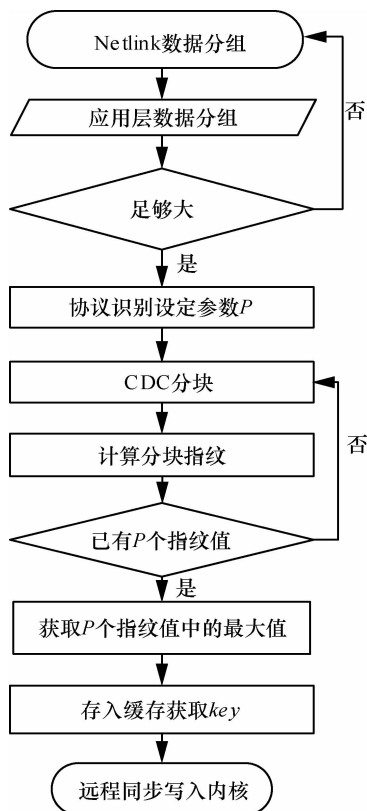


图4 数据块选取流程

- (1) 获取来自内核层的数据报文;
- (2) 如果报文长度小于 64 byte, 将不对该报文做处理;
- (3) 通过简单的协议识别技术, 判断该数据报文的应用层协议类别, 根据不同的协议类别, 初始化数据块选取参数 P ;
- (4) 通过变长分块算法 CDC 分割每一个报文, 计算每一个 CDC 分块的 32 byte 的 Rabin 指纹, 在每 P 个连续指纹值中选取局部最大值, 存入缓存, 并获取该数据块在缓存数组中的位置, 该位置即为这个数据块所对应的 key ;
- (5) 与远程接收端同步 $\langle key, chunk \rangle$;
- (6) 把 $\langle fingerprint, key \rangle$ 发送到内核层。

4.2 协议识别

协议识别采用基于 PCRE 的匹配引擎, 应用层协议采用 L7 过滤器, PCRE 首先对选取的正则表达式进行编译, 通过调用 `pcrc_compile` 函数, 生成编译好的正则表达式的 PCRE 内部结构, 然后将获取的应用层数据分组内容部分与编译好的正则表达式进行匹配, 通过调用 `pcrc_exec` 函数, 匹配正确返回对应协议类型。

协议识别部分的结构如图 5 所示。

4.3 缓存管理

缓存空间的有限性, 决定了缓存管理在提升冗余消除

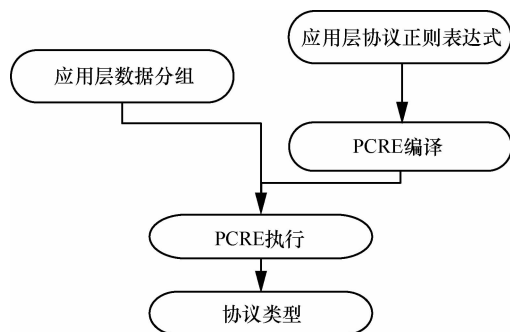


图5 协议识别结构描述

的效率方面起着决定性的作用。在之前学者的研究中, 已经提出的 FIFO 和 LRU 有优点, 但也存在着缺点, 所以这里提出了一种新的缓存置换算法。

在缓存未满的时候, 将按序把需要存入缓存的数据存起来; 当缓存已满的时候, 则触发置换算法, 置换算法的详细步骤如图 6 所示。

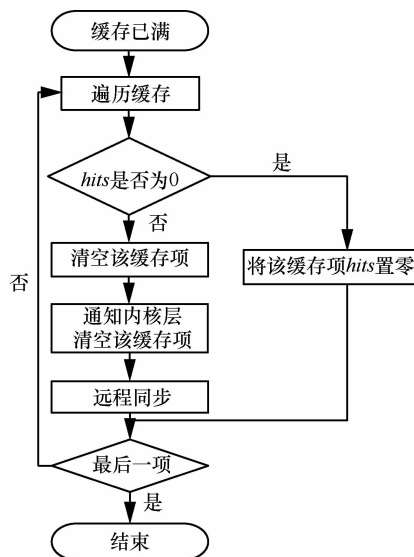


图6 缓存置换流程

4.4 内核层冗余数据消除

在本文所提出的系统中, 内核层的主要工作分为如下 3 个部分。

- 将每一个用于发送的应用层数据发往用户层;
- 接收来自用户层的缓存结构, 并把接收到的结构体存入缓存;
- 扫描即将发送的应用层数据。

与数据缓存做匹配, 把匹配到的冗余数据用独一无二的 key 来替换, 从而减少冗余数据的传输。内核层拥有和用户层完全相同的散列表, 但是却不需要缓存数组。冗余匹配通过指纹来完成, 其步骤如下:

(1)扫描应用层数据分组；

(2)通过 CDC 算法把数据分组分块,计算每一块的Rabin 指纹,与散列表中的数据做匹配,如果匹配不成功,则跳过,如果匹配成功,则用散列表中该指纹值对应的 *key* 替换原数据。

5 评估与分析

5.1 实验环境

本文采用真实的广域网传输数据进行实验,以验证 PI-RE 的算法性能。实验使用的流量是某大型局域网与广域网接口上捕捉的真实数据流量,实验数据较大,可以很好地验证算法性能。实验所用的数据分 4 次捕捉,捕捉数据的时间固定在每天的同一时间段。

为了更加直观明了地展现算法性能,将该算法与 MAXP 算法通过实验数据的统计结果进行对比,并且在数据块选取算法一致的情况下,使用所提出的缓存管理算法与 LRU 算法和 FIFO 做对比。

5.2 实验结果

将 PI-RE 算法的数据块选择算法与 MAXP 算法的性能进行对比,性能评价指标主要分两部分:带宽节约率和执行时间,带宽节约率计算如下:

带宽节约率=实际传输数据/真实数据总量×100%

5.2.1 数据选择算法比较

上述两种算法,数据块大小都选择 64 byte,缓存管理算法采用 FIFO。两种数据选择算法的比较见表 2。

表 2 MAXP 和 PI-RE 数据选择算法比较

数据分组	MAXP 冗余	PI-RE 冗余	MAXP 时间 (s)	PI-RE 时间 (s)
1	19.23%	22.02%	4 844	4 872
2	17.18%	19.57%	3 824	3 877
3	12.31%	14.72%	6 791	6 904
4	15.32%	17.57%	4 691	4 745

由表 2 可以看出,PI-RE 在冗余数据的消除方面较 MAXP 有着不错的提升,大约有 15%的性能提升。而且因为 PI-RE 系统分为内核态和用户态,所以在数据处理时间方面,并没有因为算法复杂化而有太大的差别。

5.2.2 缓存管理算法比较

表 3 中的两种算法,数据块选择算法都采用 PI-RE 算法,数据块大小选为 64 byte。

表 3 LRU 和 PI-RE 缓存管理算法比较

数据分组	LRU 冗余	PI-RE 冗余	LRU 时间 (s)	PI-RE 时间 (s)
1	22.48%	22.68%	5924	4931
2	19.89%	21.01%	4733	4128
3	14.02%	15.41%	8012	6835
4	16.91%	18.12%	5839	4911

通过两种缓存管理算法的比较可以发现,新的缓存管理算法在冗余度的提升方面大约为 10%,而在数据处理的速度上有着很大的提升,可以达到 20%。

6 结束语

本文提出的 PI-RE 算法是一种基于数据缓存的冗余数据消除策略,能够有效地减少广域网中传输的冗余数据传输,该方法灵活、可扩展。较之原有的 MAXP+LRU 有大约 20%的性能提升。针对不同的传输环境,只需要在用户态对算法进行更改,便可以达到更好的消除效果。

在下一步的工作中,将会继续对广域网中的冗余数据进行分析,抽象提取出它们的共同特点,以便更好地识别并预测,更加有效地提升冗余消除效率。

参考文献

1 Anand A, Muthukrishnan A, Ramjee A R. Redundancy in network traffic: findings and implications. Proceedings of ACM SIGMETRICS, Seattle, WA, USA, 2009:37~48

2 Squid web proxy cache. <http://www.squid-cache.org>

3 PeerApp: P2P and media caching. <http://www.peerapp.com>

4 Ziv J, Lempel A. A universal algorithm for sequential data compression. IEEE Transactions on Information Theory, 1977, 23(3): 337~343

5 Gupta A, Akella A, Seshan S, et al. Understanding and Exploiting Network Traffic Redundancy. Technical Report #1592, University of Wisconsin, 2007

6 Halepovic E, Williamson C, Ghaderi M. Low-overhead dynamic sampling for redundant traffic elimination. Journal of Communications, 2012, 7(1):28~38

7 Aronovich L, Asher R, Bachmat E, et al. The design of a similarity based deduplication system. Proceedings of the Israeli Experimental Systems Conference, Israel, 2009:1~14

8 Spring N, Wetherall D. A protocol-independent technique for eliminating redundant network traffic. Proceedings of ACM SIGCOMM, Stockholm, Sweden, 2000: 87~95