# 基于 DPI 与 DFI 的 HTTP 流归并算法

程志 [1,2]，程光 [1,2]，郭晓军 [1,2]

（1.东南大学计算机科学与工程学院，南京，211189；2.东南大学计算机网络和信息集成教育部重点实验室，南京，211189）

**摘　要：**目前的流量分类算法只能对不同类型的 HTTP 流量进行分类，但是无法将 HTTP 流量以各自所属的网页进行聚类，因此无法满足后向收费等新兴应用，本文提出了两种能将一次网页访问所产生的所有 HTTP 流进行归并的算法：基于 DPI 的 HTTP 流归并算法和基于 DFI 的 HTTP 流归并算法。前者通过解析每一个 HTTP 请求头部信息对一次网页请求产生的所有流量进行归并，适用于对精度要求较高但流量较小的网络，而后者是一种基于流量行为的应用识别技术，通过分析不同 HTTP 流在会话连接或数据流上状态的不同，从而对 HTTP 流以网页为单位进行归并，更适用于大流量下对效率较为苛刻的网络环境。实验结果表明，本文提出的两种归并算法可以很好地解决 HTTP 流量统计问题。

**关键词：**HTTP 流；归并；DPI；DFI

# Cobbling of HTTP Flow Based on DPI and DFI

Cheng Zhi [1,2], Cheng Guang[1,2], Guo Xiaojun[1,2]

(1. School of Computer Science and Engineering, Southeast University, Nanjing 211189, China;
2. Key Laboratory of Computer Networks and Information Integration (Ministry of Education),
Southeast University, Nanjing 211189, China)

**Abstract:** Currently, the flow classification algorithm can only classify HTTP traffic with different application types, and therefore unable to meet the emerging services such as reverse billing, we propose two algorithms that can cobble all HTTP traffic generated by web service: cobbling of HTTP flow based on DFI and cobbling of HTTP flow based on DPI. The former merges HTTP traffic by parsing each HTTP request header information for all the traffic generated by web browsing, which is suitable for high precision but smaller network traffic, while the latter is based on the application of the flow behavior recognition technology, by analyzing the difference of HTTP streaming in different sessions or different states, and thus merges HTTP stream by web page as a unit , which is more suitable for heavy network environments with high efficiency on heavy traffic. Experimental results show that the two proposed merging algorithm can solve the problem of HTTP traffic statistics.

**Key words:** HTTP flow; Cobbling; DPI; DFI

现阶段绝大多数 ISP（Internet Service Provider）还在提供流量计费的网络服务，因此当前 ICP（Internet Content Provider）可以替用户向 ISP 支付访问 Web 服务所产生的流量费用来吸引更多客户。如阿里巴巴已经宣布淘宝、来往、聚划算、天猫、支付宝等手机客户端应用用户可以申领它所赠送的每月 2G 定向免费流量包。但是上述这些主要针对客户端应用程序，本文提出的 HTTP 流归并算法可以对浏览器访问不同 Web 所产生的流量进行精确分类，从而统计用户访问不同网页各自所产生的流量。此外，ISP 还可以根据不同收费，对不同的 ICP 提供区分服务[1]，对于支付较高费用的 ICP，ISP 可以提高它的流量通过路由的优先级，在网络负载较高的情况下提高用户访问该网站的体验效果，从而在保留当前用户的基础上吸引更多的潜在用户。但是目前还没有能对单个页面中 HTTP 流量进行有效归并的高效算法。

虽然已经存在大量关于流量分类和识别的文献，尤其是基于应用层的流量分类[2-5]，还有针对 Web 最新技术例如 Ajax 的测量研究[6-8]，但是它们只能对不同的网络流量进行分类。Butkiewicz 等人分析现代网站在数量、类型、下载对象的大小和非起源域名例如 CDN 的复杂性[5]，发现复杂性对于用户体验的影响，此外该研究还通过机器学习[9-10]等技术将 Web 服务以它们的服务类型进行分类。但是它的

**作者简介：**程志（1987-），男，硕士研究生，Email：148133446@qq.com；程光（1973-），男，教授.

研究重点是 Web 服务分类，而不是对不同网页的 HTTP 流量进行归并。Feldmann 等人通过将 IP 与主机进行映射完成对 Web 服务的流量分类[11]，该方法已经无法适用于 Web 服务环境更为复杂的今天，例如 sina 的网页中可能包含来自 taobao 的内嵌对象，在此情况下就不能通过 IP 地址将内嵌对象与它所在的网页相关联。此外，越来越多的网站采用 CDN 加速技术，而一个 CDN 服务器可能同时为多个 ICP 服务。Ihm 提出一种 StreamStructure 算法[12]，通过 HTTP 请求头部的 Referer 将 Web 请求进行聚类，但是该文章的研究目的是 HTTP 流量的业务特性。Khandelwal 设计的 CobWeb 系统[13-14]，根据所属的网站对 HTTP 流量进行归并，但是该算法需要对报文的应用层进行解析。

针对不同的应用环境，本文提出了 2 种不同的 HTTP 流归并算法：基于 DPI 的 HTTP 流归并算法和基于 DFI 的 HTTP 流归并算法，两者都能对属于不同网页的 HTTP 流量进行归并。基于 DPI 的 HTTP 流归并算法对每一个 HTTP 请求的头部进行解析，提取头部字段 GET、Referer 以及 Host，通过将 GET 和 Host 拼接而成的 URL 与 Referer 进行匹配从而完成流之间的关联，再通过不同流之间时间等特征对主流与辅流进行识别，从而完成对一次网页请求所产生的所有流进行归并，适用于对精度要求较高但是流量较小的网络。而基于 DFI 的 HTTP 流归并算法则是从流量行为的角度，通过分析主流与辅流在会话连接与数据流上状态的细微差别，总结出规律，从而完成主流与辅流的识别与关联，更适用于大流量下对效率较为苛刻的网络环境。

# 1 基于 DPI 的 HTTP 流归并算法

## 1.1 相关概念

基于 DPI 的 HTTP 流归并算法基本思想涉及到主流，辅流等学术术语，具体阐述如下所示：

① 主流： 请求网页主页面资源的 HTTP 请求所属的流；

② 辅流： 浏览器为请求一个网页所产生的所有流中，除去主流外的所有流；

③ 父流： 若 HTTP 请求 A 中 Referer 字段值与 HTTP 请求 B 中 URL 值相同，本文称 B 请求所在的流是 A 请求所在流的父流。当一个 HTTP 流中包含多个 HTTP 请求时，只需通过流中第一个 HTTP 请求的 Referer 来确定父流；

④ 子流： 子流相对于父流而言，若 A 流是 B 流的父流，则 B 流是 A 流的子流；

⑤ 兄弟流： 同属一个父流的两个子流之间的关系就是兄弟，简称它们为兄弟流；

⑥ 关联： 本文中的关联是指通过匹配 HTTP 请求中的 GET、Host 和 Referer 字段确定流之间的关联关系。

## 1.2 算法功能

本章提出的 HTTP 流归并算法通过分析 HTTP 报文，还原 HTTP 流量的关联关系，从而完成对 HTTP 流的归并。例如访问 www.sina.com/并通过点击超链接打开 edu.sina.com.cn/与 www.xdf.cn/产生的 HTTP 流量所构建的关联关系树，如图 1 所示。

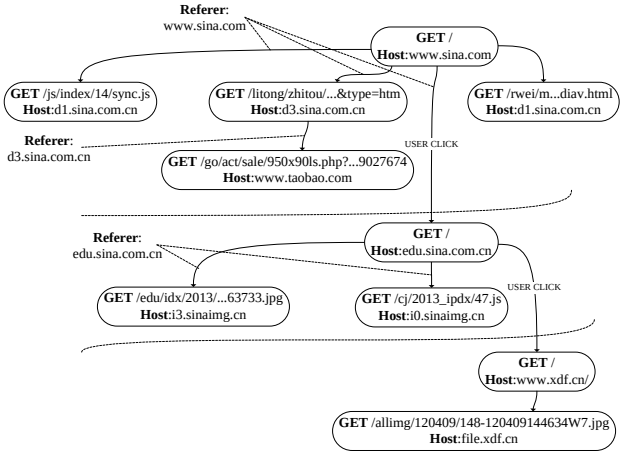


图 1 HTTP 流量关联树示例图

## 1.3 算法实现

基于 DPI 的 HTTP 流归并算法通过解析 HTTP 请求，对不同 HTTP 请求头部的几个字段进行匹配，结合各个请求之间的时间关系，完成对主流的识别以及主流与辅流的关联。一个完整的 HTTP 请求包含许多有关客户端环境和请求正文的有用信息，但是本算法只需处理 GET、Host 和 Referer 三个字段。

### 1.3.1 流之间的关联方法

本文通过匹配 HTTP 请求头部的 Referer 以及 URL 对流进行关联，Referer 是 HTTP 协议 Header 的一部分，当浏览器向 Web 服务器发送请求的时候，通过在请求头部记录 Referer 通知服务器该请求的源端页面。HTTP1.1 协议中浏览器可以在一个 TCP

连接上向服务器发送多个 HTTP 请求。一般情况下，虽然一个 HTTP 请求头部都会包含 Referer 字段，只需处理每条流中第一个 HTTP 请求的 Referer，因为当一条流中出现多个 HTTP 请求时，只需通过第一个 HTTP 请求的 Referer 对其进行关联，而无需关联同一个流中的后续 HTTP 请求，通过这种技巧可以有效提高程序的执行效率。

### 1.3.2 主流识别

本文通过兄弟流以及父子流之间的时间特征完成主流的识别。当浏览器请求一个页面时，正常情况下它会首先建立一条 TCP 连接并在此连接上发送第一个 HTTP 请求，当收到该 HTTP 请求对应的 HTTP 响应报文并解析后，浏览器随即建立多条 TCP 连接并通过这些连接发送后续 HTTP 请求（至少 3 个）用于获取该页面内嵌对象资源。如图 2 所示，在 hao123 的主页中点击超链接新浪网（www.sina.com.cn）后，浏览器首先会为新浪网建立一条新的主流并通过它发送 HTTP 请求（$T_{G0}$、$T_{G1}$、$T_{G2}$、$T_{G3}$、$T_{G4}$为 HTTP 请求的时戳），当服务器返回该 HTTP 请求对应的响应后（该响应对应的开始时间与结束时间分别为$T_{R1}$与$T_{R2}$），浏览器处理 HTTP 响应的载荷并解析出内嵌对象资源 URL，通过这些 URL 的 Host 建立多条 TCP 连接并请求内嵌对象资源，本文只考虑前 3 个 Referer 指向主页 URL 的 HTTP 请求，时间分别记作$T_{G2}$、$T_{G3}$与$T_{G4}$。

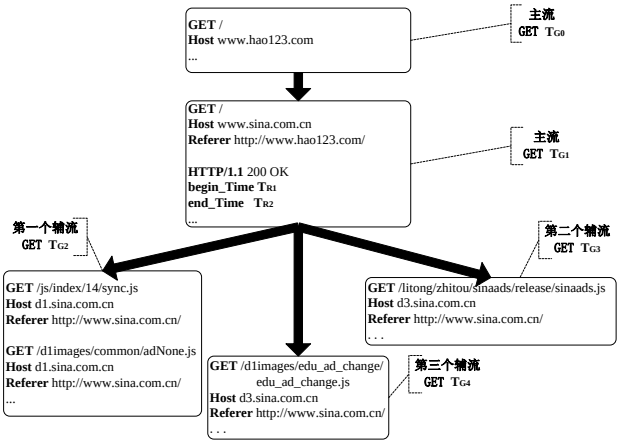


图 2 主流与辅流关系示例图

浏览器在请求一个网页时可分两种情况：

① 用户在已打开的页面上点击链接（Href）产生新的页面；

② 用户在浏览器的地址栏中输入 URL 打开新的页面。

因此，若$T_{G1}$对应的 HTTP 请求所在的流为主流，那么$T_{G0}$、$T_{G1}$、$T_{G2}$、$T_{G3}$、$T_{G4}$、$T_{R1}$和$T_{R2}$之间的关系满足如下条件：

① $\max(T_{G2}, T_{G3}, T_{G4}) - T_{R2} < Tthresh1$

② $T_{G1}-T_{G0} > Tthresh2 \,||\, Referer == NULL$

Tthresh1 取 950 毫秒，Tthresh2 取 5 秒。Tthresh1 取值通过对 Alexa.com 公布的中国百大访问量网站 HTTP 流量分析后得出。

此外本文还通过对 URL 的分析来提高识别主流的精度。主流是请求网页主页面资源的 HTTP 请求所在的流，所以若一个流中第一个 HTTP 请求所访问的资源是图片、Flash、Css 等，这些 HTTP 请求中 URL 会包含.css、.jpg、.swf 等表示文件类型的字段，且这个流一定不是主流。

Algorithm1：主流识别算法

```
for each flow's request req
BEGIN
if (req.url is not embedded file-type)
    req_ht.insert(req.url) and req_ht(req.url).subGetNum = 0
if (req.referer is not NULL and req.time - req_ht(req.referer).eTime <
  Tthresh1)
    req_ht(req.referer).subGetNum++;
if (req_ht(req.referer).subGetNum eq 3)
    if (req_ht(req.referer).referer eq NULL)
            req_ht(req.referer) is mainflow
    else if (req_ht(req.referer).sTime- req_ht(req_ht(req.referer).referer).sTim
          > Tthresh2)
            req_ht(req.referer) is mainflow
END
```

图 3 主流识别算法

图 3 给出了主流识别算法，其中哈希表 req_ht 保存所有 URL 中不包含表示文件类型字段的 HTTP 请求。req_ht(req.referer)表示通过请求 req 中 Referer 值在哈希表 req_ht 找出相应的 HTTP 请求并返回该请求。

### 1.3.3 异常处理

在实际中存在一些异常情况，会对主流的识别造成一定的影响，主要有重定向和广告流量。下面分别对此两种情况给出处理方案：

1）主页重定向

当在地址栏中输入地址 URL1，在浏览器中却打开重定向后的地址 URL2 时，地址为 URL1 的 HTTP 请求虽为浏览器为访问该页面所发送的第一个 HTTP 请求，但本文将地址为 URL2 的 HTTP 请求所属的流定义为主流。算法对重定向的处理如下：若 A 流中 HTTP 响应报文状态码为 301 或 302 时认为它被重定向，这个 HTTP 响应报文头部中 Location 字段的值就是被重定向后的地址 URL2，本文将这个地址存储在一个称为 redirect_URL 的单链表中。对于 B 流，在处理它的 HTTP 请求时，首先判断它访问的 URL（这里为 URL2）在 redirect_URL 链表中是否存在相同的项，若存在则表示 B 流中 HTTP 请求是另一个流（这里是 A 流）中 HTTP 请求重定向而来，那么将 B 流定义为 A 流的主流。

2）广告流量

通过上述步骤，基本完成对主流与辅流的识别，并将辅流正确地关联到各自所属的主流上。但是，网页中往往还包含第三方内嵌对象资源，例如广告、微博分享等，它们的资源并不存储在当前网页所属的服务器上，而且浏览器会在加载完当前页面的其它内嵌对象资源后才请求这些第三方资源，这就导致这些第三方资源的流量特征与主流的特征相仿而被程序错误地辨别为主流。

这些第三方资源（主要是广告），它们第一个 HTTP 请求中 Referer 一般较长，例如访问 http://www.sina.com.cn/时浏览器会请求内嵌广告资源，该广告会被程序错误地识别为主流，该广告主流中第一个 HTTP 请求的 Referer 的格式：http://host/...www.sina.com.cn...。

对于这些广告流，在它们的 Referer 中会内嵌它所属主流的 m_URL（这里为了区分，将主流的 URL 都称为 m_URL），如本例中的 http://www.sina.com.cn/，因此每当程序通过之前提出的方法确定为主流后，还需要判断它是否为广告流，遍历 mainURL_list 中存储的所有已被识别的主流 m_URL，若存在一个主流 m_URL 被当前 Referer 中的 Path 所包含（Referer = Host + Path），则认为当前流只是一个特征类似主流的广告流。

此外，绝大多数广告流量来自淘宝、京东、百度等大型互联网企业，本文还通过建立黑名单对广告流量进行识别。

# 2 基于 DFI 的 HTTP 流归并算法

基于 DFI 的 HTTP 流归并算法从流的角度，分析主流与辅流之间的差异并提炼出规则，从而识别主流与辅流并在此基础上完成辅流与主流的关联。本文针对两种不同的场景：主流在传输 HTTP 请求时是否复用 TCP 连接，提出了两套识别规则。

## 2.1 流中多事务场景的主流识别算法

由于 HTTP 1.1 支持长连接（Persistent Connection），在一个 TCP 连接上可以传送多个 HTTP 请求和响应，减少了建立和关闭连接的消耗和延迟。

在多事务场景中主流将满足如下条件：

① 主流服务器端 IP 为新 IP；

② 主流中承载主页面资源的 HTTP 响应包含 TCP 报文段数目应大于等于阈值 3，且主流中应至少包含 2 个 HTTP 请求；

③ 从主流 HTTP 响应第一个 TCP 报文段下载开始后 20ms，到完成全部下载后 950ms 内应至少有 3 个相关 HTTP 请求被加载，且这些 HTTP 请求对应的服务器端 IP 为新 IP 或与主流具有相同的 IP；

④ 若主流第一个 HTTP 响应只由一个 TCP 报文段构成，那么主流中第二个 HTTP 请求与响应必须满足上述 3 个条件；

⑤ 当前主流与前一个主流间隔时间应大于 5 秒。

## 2.2 流中单事务场景的主流识别算法

对于部分 IP 对应的主流，浏览器与服务器只保持短暂的连接，浏览器的每次请求都需要与服务器建立一个新的 TCP 连接，服务器完成请求处理后立即断开 TCP 连接，服务器不跟踪每个客户也不记录过去的请求，从特征来看这些主流非常符合 HTTP 1.0 协议的特征。此外，由于大多数网页的流量都比较小，一次 TCP 连接很少能通过 slow-start 区，不利于提高带宽利用率，因此符合该特征的主流并不多，但在实验过程中还是发现部分流量符合该特征的主流。

单事务场景中主流应满足如下条件：

① 在一定的时间区间内，一个 IP 相关的 3 个 TCP 连接中 HTTP 请求不超过 1 个，主流为该 IP

第一个 HTTP 请求所属的流；

② 主流服务器端 IP 为新 IP；

③ 从主流 HTTP 响应第一个 TCP 报文段下载开始后 20ms，到完成全部下载后 950ms 内应至少有 3 个相关 HTTP 请求被加载，且这些 HTTP 请求对应的服务器端 IP 为新 IP 或与主流具有相同的 IP；

④ 当前主流与前一个主流间隔时间应大于 5 秒。

### 2.3 辅流与主流的关联

此节给出辅流与主流关联的方法，通过 2.1 节和 2.2 节中给出的算法完成主流与辅流的识别后对辅流与主流进行关联。

若一条辅流同时满足下述条件时，将它关联到离它最近的主流 mainflow 上：

① 辅流服务器端 IP 为新 IP；

② 辅流中第一 HTTP 请求时戳大于 mainflow 中第一个响应的开始时戳+浏览器处理时间。

否则将该辅流关联到离它第二近的主流上。此外还设定超时时间，只对每个主流开始之后 32 秒内的辅流进行关联。因为浏览器不必等收到服务器端返回的完整 HTTP 响应后才对它进行解析，因此，若一条流中第一个 HTTP 请求时戳大于主流第一个响应中首个 TCP 报文段时戳，就可以认为它是该主流的辅流。考虑浏览器处理时间是因为浏览器在收到 HTTP 响应的分组后进行解析才能发送后续辅流的请求，本文将这段时间也考虑进来。本关联算法只通过每一条流中第一个 HTTP 请求对流进行关联，对于之后的 HTTP 请求，由于它所在的流已经通过首个 HTTP 请求被关联，所以不必再被关联。

## 3 实验结果与分析

### 3.1 实验数据与评价指标

本文的实验数据来自访问 20 个在 alex 排名较高的网站产生的流量，数据大小约为 9G，同时记录所访问页面的 URL，对主流进行手动标记。

为了评价 HTTP 流归并算法的性能，本文选用 3 种评价指标：

① 查准率（Precision）：假设 N 表示被算法标记为主流的样本数目，T 表示被正确标记为主流的样本数目。查准率$R_{TP} = T/N$；

② 查全率（Recall）：假设 M 为实际样本中主流数目，T 表示被正确标记为主流的样本数目。查全率$R_{FN} = T/M$；

③ 关联字节比率：关联字节比率表示归并结果中每个网页的总字节数与对应网页实际字节数的比值。

查准率和查全率体现了识别方法针对主流的识别效果，关联字节比率体现了辅流与主流关联上的准确率。

### 3.2 实验结果分析

图 4 与图 5 分别给出 20 个数据集主流识别的查准率与查全率。从图 4 与图 5 可见，基于 DPI 的 HTTP 流归并算法具有较高的查全率与查准率，尤其在部分数据集上的实验结果明显超越基于 DFI 的 HTTP 流归并算法。因为 DPI 方法可以解析 HTTP 请求头部，通过 Referer 字段确定流之间的关系，因此该方法的识别结果具有较高的正确率，而 DFI 方法只是根据 IP 与时间等特征确定流之间的相关性，导致正确率不高。又因为两种方法对识别主流的思想基本一致，因此在某些数据集上两种方法的结果表现比较相近。

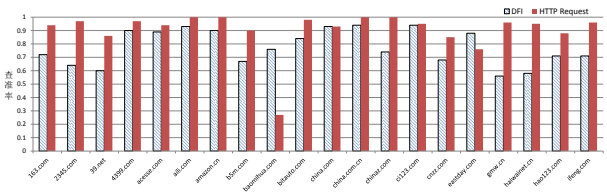


图 4 查准率

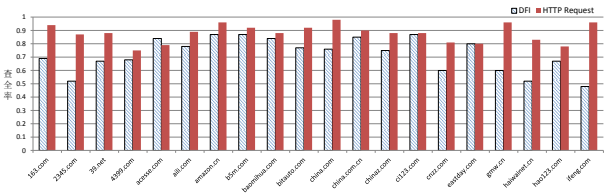


图 5 查全率

由于基于 DPI 的归并算法通过 Referer 对流进行关联，所以若一个页面的主流能被正确识别，则可以将该页面 HTTP 流量的归并结果作为该页面的实际流量。图 6 给出 DFI 算法归并结果与网页实际字节数的比值。

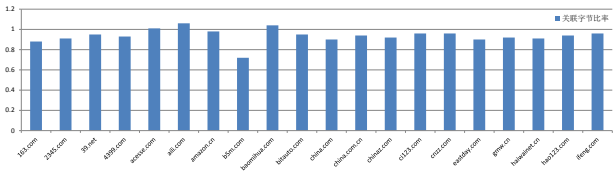


图 6 关联字节比率

## 4 结束语

针对网络后向收费以及路由加速应用，本文提出了两种 HTTP 流归并算法：基于 DPI 的 HTTP 流归并算法和基于 DFI 的 HTTP 流归并算法。基于 DPI 的 HTTP 流归并算法通过解析 HTTP 请求的内容对一次网页请求产生的所有流量进行归并，而基于 DFI 的 HTTP 流归并算法通过总结不同 HTTP 流在会话连接或数据流上状态的规律，对 HTTP 流进行归并。实验结果表明两种算法均获得了较好的归并结果。下一步将在提高算法归并精度尤其是 DFI 算法，展开研究工作。

## 参考文献

[1] Allot. http://www.allot.com.

[2] T. Karagiannis, K. Papagiannaki, and M. Faloutsos.Blinc: Multilevel traffic classification in the dark. In ACM SIGCOMM, 2005.

[3] J. Ma, K. Levchenko, C. Kreibich, S. Savage, and G. M. Voelker. Unexpected Means of Protocol Inference.pages 313–326, October 2006.

[4] M. Roughan, S. Sen, O. Spatscheck, and N. Duffield. Class-of-Service Mapping for QoS: A Statistical Signature-based Approach to IP Traffic Classification. In ACM IMC, 2004.

[5] M. Butkiewicz, H. V. Madhyastha, and V. Sekar. Understanding website complexity: measurements, metrics, and implications. In IMC, 2011.

[6] E. Kiciman and B. Livshits.Ajaxscope: a platform for remotely monitoring the client-side behavior of web 2.0 applications. SIGOPS Operating System Review, 41(6):17–30, 2007.

[7] M. Lee, R. R. Kompella, and S. Singh. Active measurement system for high-fidelity characterization of modern cloud applications.In Proceedings of USENIX Conference on Web Applications, 2010.

[8] F. Schneider, S. Agarwal, T. Alpcan, and A. Feldmann. The New Web: Characterizing AJAX Traffic. In International Conference on Passive and Active Network Measurement, April 2008.

[9] I. Katakis, G. Meditskos, G. Tsoumakas, N. Bassiliades, and I. Vlahavas. On the combination of textual and semantic descriptions for automated semantic web service classification. In AIAI, 2009.

[10] Oldham N, Thomas C, Sheth A, et al. Meteor-s web service annotation framework with machine learning classification[M]//Semantic Web services and Web process composition. Springer Berlin Heidelberg, 2005: 137-146.

[11] A. Feldmann, N. Kammenhuber, O. Maennel, B. Maggs, R. D. Prisco, and R. Sundaram. A methodology for estimating interdomain web traffic demand. Oct. 2004.

[12] S. Ihm and V. S. Pai.Towards understanding modern web traffic.In IMC, Nov. 2011.

[13] Khandelwal H, Hao F, Mukherjee S, et al. CobWeb: A System for Automated In-Network Cobbling of Web Service Traffic[J]. 2012.

[14] Khandelwal H, Hao F, Mukherjee S, et al. CobWeb: In-network cobbling of web traffic[C]//IFIP Networking Conference, 2013. IEEE, 2013: 1-9.