目 录

1 web 代理服务器的实现

这份报告首先描述了 web 代理服务器实现了什么功能，然后讲述如何去实现这个 web 代理服务器。阅读者只要有 python 编程和 socket 编程的基础，就可以根据这篇文档写一个代理服务器。

1.1 web 代理服务器的功能

1.1.1 HTTP 请求转发

这个 web 代理服务器实现了 HTTP 请求转发，也就是当我的代理服务器从一个浏览器接收到对某对象的 HTTP 请求时，它生成对相同对象的一个新 HTTP 请求，并向初始服务器发送。

如下图所示：

```
接收来自： ('10.180.161.60', 49482) 的HTTP请求
请求内容如下：
----------------------------------------

GET http://cn.bing.com/ HTTP/1.1
Host: cn.bing.com
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Proxy-Connection: keep-alive
Upgrade-Insecure-Requests: 1
Cookie: MUIDB=1D8F7AD837F56AD92BCD739E36546B0E; SRCHUID=V=2&GUID=C4429C13B10745A9A23
610CE328A06E9; SRCHHPGUSR=CW=1274&CH=735&DPR=2&UTC=480; WLS=TS=63628679975; _SS=SID=
2B889D2F2C1A6B970ACB97412DBB6AD1&HV=1493083178&bIm=859676; _EDGE_S=mkt=zh-cn&SID=2B8
89D2F2C1A6B970ACB97412DBB6AD1; ipv6=hit=1; _IFAV=COUNT=0; MUID=1D8F7AD837F56AD92BCD7
39E36546B0E; _ITAB=STAB=TR; SRCHD=AF=APMCS1; SRCHUSR=DOB=20160702; _EDGE_V=1
User-Agent: Mozilla/5.0 (Macintosh; Intel Mac OS X 10_12_4) AppleWebKit/603.1.30 (KH
TML, like Gecko) Version/10.1 Safari/603.1.30
Accept-Language: zh-cn
Accept-Encoding: gzip, deflate
Connection: keep-alive



----------------------------------------
将HTTP请求转发给 cn.bing.com : 80 初始服务器
```

1.1.2 回送 HTTP 内容替代

这个 web 代理服务器还实现了回送 HTTP 内容替代，也就是代理服务器从初始服务器接收到具有该对象的 HTTP 响应时，它生成一个包括该对象的新 HTTP 响应，并发送给该客户。

如下图所示：

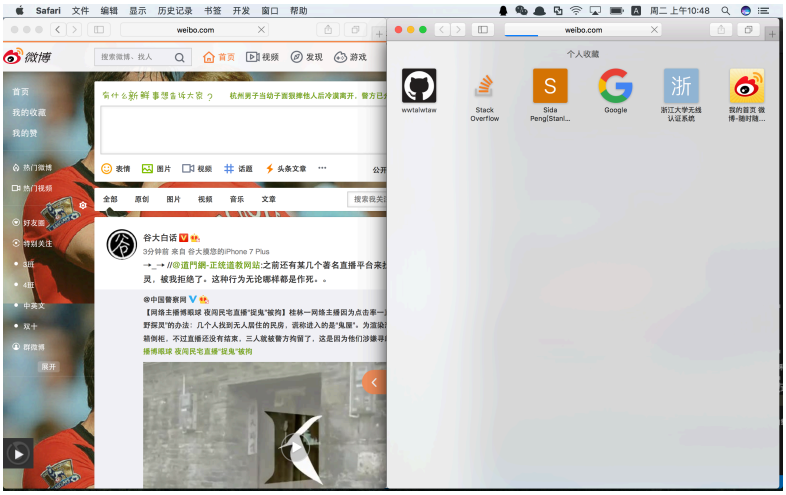
```
接收来自：('cn.bing.com', 80) 的具有该对象的HTTP响应
------------------------------------------

HTTP/1.1 200 OK
Cache-Control: no-cache
Pragma: no-cache
Content-Length: 4892
Content-Type: application/x-javascript
Expires: -1
Vary: Accept-Encoding
Server: Microsoft-IIS/8.5
X-MSEdge-Ref: Ref A: 629363F0C13D4FB4B09E0ECBC59A1CDB Ref B: BJ1SCHEDGE0123 Ref C: Mon Apr 24 19:33:44 2017 P
ST
Date: Tue, 25 Apr 2017 02:33:44 GMT

(function(){var n=".sc_pc{position:absolute;padding:0;top:14px;font-size:92%;margin:.82em 0 0 -18px;visibilit
y:hidden;min-width:16em;width:auto}.sc_pc .sc_hl1{margin-right:0;color:#acacac}.sc_pc>.sc_hl1{padding:10px 8p
x}.sw_tb .sc_hl1 li .sc_hl1 li{margin:0}.hp_hor_hdr .sw_tb .sc_pc .sc_hl1 li a{margin:0 9px}.sc_pc .sc_pcC a{
word-wrap:break-word;white-space:normal;width:14.5em}.sc_pc .sc_hl1 li a{line-height:18px !important;white-sp
ace:normal}.sc_pc .sc_hl1 a:hover{text-decoration:none;color:#fff}#hp_sw_hdr .sc_pc h3{display:block}.sc_pc l
i{padding:.15em}.sc_sct{background:#000;left:0;top:0;position:absolute;z-index:-1;width:100%;height:100% !imp
ortant}#hp_sw_hdr .sw_tb .sc_pc,#hp_sw_hdr .sw_tb .sc_pc div,#hp_sw_hdr .sw_tb .sc_pc li{display:block}.sc_sc
t{filter:alpha(opacity=70);opacity:.7}.sc_pcC{float:left;width:16.6em}.sc_pc h3,.sc_pc .scphdr{font-size:18px
;color:#fff;margin:0 .8em;color:#fff;font-weight:normal;padding:.3em .2em}.sc_pcA{padding:2.1em .2em 0;paddin
g-left:5px;clear:both}.sc_pcA A{margin:0 .75em}.sc_pc a{display:inline-block;padding:0}.sc_pc a:hover{text-de
coration:underline}.sc_pc .sc_hl1 a,.sc_pc a,.sc_pcA{color:#acacac;font-family:Arial,Helvetica,Sans-Serif;fon
t-size:12px;text-transform:none}#history.sc_C2{width:34em}#history li a{margin:8px 18px}";sj_ic(n)})(),sc_Pop
upFetcher=function(){function s(){hp_hasclass(i,p)||(i.className+=" "+p),sj_ue(n,a,s,!1),sj_ue(n,c,s,!1)}func
tion f(n){var e;k(n,!1),t&&t!=n&&w(t),t=n;var f=n.pop,o=sj_go(n,nt),u,s=_ge(pt),r=_d.body.clientWidth-s.offse
tWidth,i=n.firstChild&&n.firstChild.offsetLeft-n.offsetLeft||0;i<0&&(i=n.firstChild.offsetLeft,r<0&&(r=0)),ft
?(i===0&&n.firstChild&&(i=n.offsetWidth-n.firstChild.offsetWidth),e=sj_go(_ge("sc_hdu"),nt),u=o-e,f.style.lef
t=u+n.offsetWidth-f.offsetWidth-i+h):(u=o-(r>0?r/2:0),f.style.left=u+i+h),sj_evt.fire("onSPHover","scPop"+_sc
opeRef[f.id]),sb_i6?d(n):n.delayfadeInInt=sb_st(sj_wf(d,n,!0),b)}function r(n,t){var i=sj_mo(t);sj_we(i,n)||(
k(n,!0),n.delayfadeOutInt=sb_st(sj_wf(w,n),b))}function tt(n,i){var c,a,l,h;if(i||(i=window.event),c=i.which|
|i.keyCode,c){c!==u.tab&&c!==u.shift&&sj_pd(i);var s=n.firstChild,e=n.pop,o=_d.activeElement;if(o&&s)switch(c
){case u.escape:t==n&&(s.focus(),r(n,i));break;case u.enter:e&&t!=n?f(n,i):o.click();break;case u.tab:i.shift
Key?e&&o==s&&r(n,i):e&&o!=s&&(h=v(e,o),h||r(n,i));break;case u.up:t==n?o==s?r(n,i):(a=rt(e,o),a?a.focus():s.f
ocus()):e&&f(n,i);break;case u.down:t==n?o==s&&e?(l=v(n,s),l&&l.focus()):(h=v(e,o),h&&h.focus()):e

------------------------------------------
将HTTP响应回送给 ('10.180.161.60', 49926) 客户
```

1.1.3 支持多用户访问功能

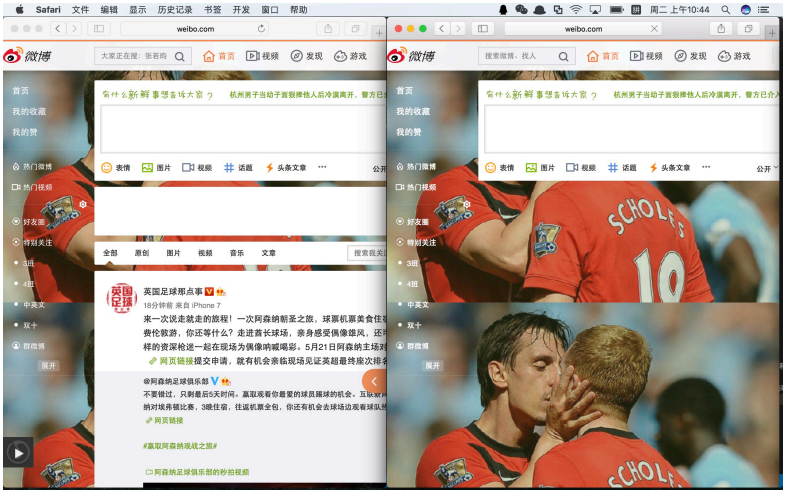
这个 web 代理服务器支持多线程，也就是支持多用户访问功能。这个功能比较难描述，我主要讲一下单线程代理服务器和多线程代理服务器的区别。

如果是单线程代理服务器，我在请求一个网页的时候，如果这个网页还未加载完成，那么随即请求另一个网页就无法成功。而如果是多线程代理服务器，就可以同时请求多个网页。

单线程代理服务器的效果如下图，也就是左边的网页还未加载结束，而同时请求右边的网页，右边的网页就不会有任何显示：



多线程代理服务器的效果如下图，也就是左边的网页还未加载结束，而同时请求右边的网页，右边的网页也可以开始加载：



1.2 web 代理服务器的实现思路

这部分只是讲解 web 代理服务器功能实现的思路，一些具体的细节仍需要参见完整的源码，源码在附件中。

1.2.1 如何实现 HTTP 请求转发

实现思路如下：

1. 建立一个代理 socket，监听 tcp 连接请求。
2. 建立 tcp 连接以后，从客户 socket 获得请求报文。
3. 从请求报文中提取初始服务器，并建立与初始服务器的连接。
4. 将 HTTP 请求转发给初始服务器。

建立一个代理 socket，监听 tcp 连接请求：

```
    # 建立一个代理socket
    port = 12000
    proxy_socket = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
    proxy_socket.bind(('', port))
    proxy_socket.listen(5)
    while 1:
        # 监听tcp连接请求
        client_socket, client_addr = proxy_socket.accept()
        # 开始处理来自客户的请求
        proxy_thread(client_socket, client_addr)
```

建立 tcp 连接以后，从客户 socket 获得请求报文：

```
    request = client_socket.recv(999999)
```

从请求报文中提取初始服务器，并建立与初始服务器的连接：

```
    # 提取初始服务器的主机地址和端口
    host, port = get_host_and_port(request)
    server_socket = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
    # 建立与初始服务器的连接
    server_socket.connect((host, port))
```

这里 get_host_and_port() 是我另外写的一个函数，用于提取请求报文中初始服务器的主机地址和端口。一个典型的 HTTP 请求报文如下所示：

```
GET http://cn.bing.com/ HTTP/1.1
Host: cn.bing.com
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Proxy-Connection: keep-alive
Upgrade-Insecure-Requests: 1
```

```
Accept-Language: zh-cn
Accept-Encoding: gzip, deflate
Connection: keep-alive
```

初始服务器的主机地址和端口存放在第一行，如果第一行中没有声明端口，说明初始服务器的 web 服务器运行在 80 端口。我们可以使用如下函数提取：

```
    def get_host_and_port(request):
        first_line = request.split('\n')[0]
        url = first_line.split(' ')[1]

        http_pos = url.find("://")
        if http_pos == -1:
            temp = url
        else:
            temp = url[(http_pos+3):]

        host_pos = temp.find("/")
        if host_pos == -1:
            host_pos = len(temp)

        port_pos = temp.find(":")

        if (port_pos == -1 or host_pos < port_pos):
            port = 80
            host = temp[:host_pos]
        else:
            port = int((temp[(port_pos+1):])[:host_pos-port_pos-1])
            host = temp[:port_pos]

        return host, port
```

最后将 HTTP 请求转发给初始服务器：

```
    server_socket.send(request)
```

1.2.2 如何实现回送 HTTP 内容替代

实现思路如下：

1. 首先接收来自初始服务器的数据。
2. 然后将数据发送给客户。
3. 最后关闭客户器端 socket 和服务器端 socket。

接收来自初始服务器的数据:

```
    data = server_socket.recv(999999)
```

将数据发送给客户：

```
1     client_socket.send(data)
```

最后关闭客户器端 socket 和服务器端 socket:

```
1     server_socket.close()
2     client_socket.close()
```

1.2.3 如何实现支持多用户访问功能

这个只要使用多线程编程就可以了，只要有客户请求，我们就开一个新线程处理这个请求，代码如下:

```
1     while 1:
2         client_socket, client_addr = proxy_socket.accept()
3         thread.start_new_thread(proxy_thread, (client_socket, client_addr))
```