

文章编号:1001-9081(2005)07-1688-04

TURN 服务器原型系统的设计与实现

李鸿彬^{1,2}, 杨雪华^{1,2}, 雷为民²

(1. 中国科学院 研究生院, 北京 100039; 2. 中国科学院 沈阳计算技术研究所, 辽宁 沈阳 110004)
(lihongbin_sict@yahoo.com.cn)

摘 要: TURN 协议是一种穿越对称 NAT 的技术。以 RFC3489 技术为基础, 对 TURN 协议草案进行了深入研究与分析, 并在此基础上, 改进和简化了草案中存放动态分配地址的地址映射表结构, 设计了 TURN 技术的工作方式和应用模型。然后, 借鉴了 STUN 方式的设计思想, 设计并实现了 TURN 服务器原型系统, 解决了 SIP UA 在 STUN 等方式下不能穿越对称性 NAT 问题。

关键词: 下一代网络; UDP 对 NAT 的简单穿越; 通过中继方式穿越 NAT; 实时传输协议; 中继
中图分类号: TP393.02 **文献标识码:** A

Design and implementation of a prototype system of TURN server

LI Hong-bin^{1,2}, YANG Xue-hua^{1,2}, LEI Wei-min²

(1. Graduate School, Chinese Academy of Sciences, Beijing 100039, China;
2. Shenyang Institute of Computing Technology, Chinese Academy of Sciences, Shenyang Liaoning 110004, China)

Abstract: TURN protocol is a technique for simple traversal of UDP through NAT. On the basis of RFC3489, the draft for TURN protocol was researched and analyzed in detail. Its address translation table was modified and simplified, which stored dynamic allocated addresses, and the working mode and application model of TURN technology were designed. Then, consulted STUN design ideas, a prototype system of TURN server was designed and implemented, which solved the problem that SIP UA can't traverse symmetric NAT by STUN.

Key words: NGN; STUN; TURN; RTP; relay

0 引言

目前 NGN 网络逐步从试验走向商用, 大量的企业网和驻地网基本上都采用私有 IP 地址通过出口的 NAT/FW 接入到公网。而在企业网和驻地网上, 要求承载语音、多媒体和视频等业务越来越普遍。如果终端用户处在 NAT 之后, 由于 NAT 仅对 IP 包的地址及端口号进行转换, 而 H.248, SIP^[1] 等协议真正的媒体连接信息是放在 SDP^[2] (即 IP 包的负载) 中传递的, 这部分私网地址无法被 NAT 映射成公网地址传到对方用户, 所以媒体流是无法真正建立起来的; 并且 NAT 如何保持记录的会话地址转换直到通话结束才被删除, 这都是目前这一领域有待解决的问题。

针对以上提到的问题, 目前的解决方案主要有如下几种: NAT/ALG, MIDCOM, STUN^[3], TURN^[4] 和 Full Proxy。由于采用 TURN 方式穿越 NAT 还没有真正的标准, 因此, 本文对于 TURN 协议草案进行了深入的研究与分析, 并在此基础上, 改进和简化了草案中存放动态分配地址 (IP 地址和端口) 的地址映射表结构, 设计了 TURN 技术的工作方式和应用模型。并在 Linux 下针对 SIP 电话业务设计和实现了 TURN 服务器原型系统, 解决了 SIP UA 在 STUN 等方式下不能穿越对称性 NAT 的问题。

1 工作方式的设计

TURN 方式解决 NAT 问题的设计思路是: 对于基于私网接入用户来说, 它们可以通过某种机制预先得到其私有地址

对应在公网的地址 (STUN 方式得到的地址为出口 NAT 上的地址, TURN 方式得到地址为 TURN Server 上的地址), 然后将报文负载中描述的地址信息直接填写为公网 TURN Server 上的地址。这样报文负载中的内容在经过 NAT 时就无需被修改了, 只需按普通 NAT 流程转换报文头的 IP 地址即可, 负载中的 IP 地址信息和报文头地址信息又是一致的, TURN Server 对后续的报文根据分配的地址和端口信息作地址变换后 Relay 转发。TURN 协议的设计就是基于此思路来解决应用层地址的转换问题。

在该设计中, 将 TURN 设计为一种简单的客户机-服务器应用层协议。TURN 客户机首先查找 TURN 服务器的地址, 当找到服务器地址后, 就向服务器发送一个分配请求消息, 假设分配被服务器授权并没有被欺诈, 则服务器为该客户机分配一个公网上的 IP 地址和端口 (这个端口将被打开, 并在有效时间内一直处于被监听状态), 并将这个传输地址放入到分配响应的某一属性域中, 然后返回给客户机。同时, 监听这个端口。假设在固定的时间内没有通信或预分配的请求到达, 则释放这个端口。这样, 其他设备可以重新使用该端口。

2 应用模型

TURN 的全称为 Traversal Using Relay NAT, 即通过 Relay 方式穿越 NAT, 它的应用模型如图 1。

TURN 应用模型通过分配 TURN Server 的 IP 地址和端口作为客户端对外的可见地址和端口, 即私网用户发出的报文都要经过 TURN Server 进行 Relay 转发。这种方式的应用模

收稿日期: 2004-12-17; 修订日期: 2005-03-10

作者简介: 李鸿彬 (1973-), 男, 河北邢台人, 硕士研究生, 主要研究方向: IP 通信、软交换; 杨雪华 (1978-), 女, 辽宁营口人, 硕士研究生, 主要研究方向: 软件工程、代码重用; 雷为民 (1969-), 男, 山西平遥人, 研究员, 博士, 主要研究方向: IP 通信、软交换。

型除了具有 STUN 方式的优点外,还解决了 STUN 应用无法穿透对称 NAT(Symmetric NAT)以及类似的 Firewall 设备的缺陷,即无论企业网/驻地网出口为哪种类型的 NAT/FW,都可以实现 NAT 的穿透,TURN 还支持基于 TCP 的应用,如 H.323 协议。此外,TURN Server 控制分配地址和端口,能分配 RTP/RTCP^[5,6] 地址对(RTCP 端口号为 RTP 端口号加 1)作为本端客户的可见地址,避免了 STUN 应用模型下出口 NAT 对 RTP/RTCP 地址端口号的任意分配,使得客户端无法收到发送端发过来得 RTCP 报文(发送端发送 RTCP 报文时,目的端口号缺省按 RTP 端口号加 1 发送)。

考虑到安全问题,本文在 TURN 协议中采用了和 STUN 协议类似的安全方法,但是,客户机必须具有和服务器通信的权限。为了和 TURN 服务器通信,TURN 客户机必须先通过服务器验证,得到服务器的鉴定;并且,拥有服务器给其分配临时的用户名和密码后,才允许发送分配请求。客户机通过接收共享密码响应中的 NONCE 和 REALM 域,采用 MD5 算法^[7]产生证书,并将证书传给服务器。若证书正确,则服务器为客户机临时分配一个权限(临时的用户名和密码),客户机使用该权限构成分配请求的完整性属性。具体说明见信令处理中的 Shared Secret Request 部分。

TURN 方式的局限性在于需要终端支持 TURN Client,这一点同 STUN 一样对网络终端有要求。此外,所有报文都必须经过 TURN Server 转发,增加了包的延迟和丢包的可能性。

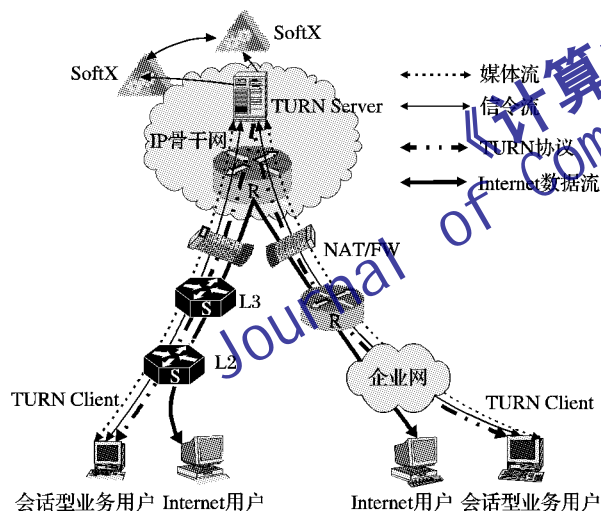


图1 TURN 应用模型

3 TURN 协议的设计

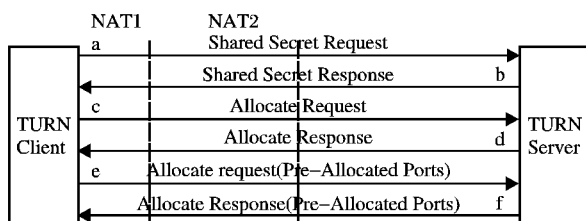


图2 TURN 信令流程图

在该设计中,本文对于 TURN 方式的管理采用了带外管理,客户机发出请求,服务器使用一个固定的端口产生响应,并发送 TURN 响应消息给客户机。具体的信令流程见图 2(字母表示先后顺序)。对于每一对话,使用 TURN 服务器分配的地址(IP 地址和端口)接收。对于 UDP 数据报,客户端(Client)将数据从源地址发送到服务器(Server)为其分配

的 IP 地址和端口上;然后,服务器将数据转发到映射表中的目的地址。

TURN 协议定义了三类消息,定义如下:

Allocate Request	0x0003
Allocate Response	0x0103
Allocate Error Response	0x0113

另外还使用了三类消息:

Shared Secret Request	0x0002
Shared Secret Response	0x0102
Shared Secret Error Response	0x0112

消息的长度是以字节为单位的,但不包括 20 个字节的消息头。事务 ID 号是一个 128 位的标识符,主要用于标识不同的请求响应对。消息头的格式如下:

TURN Message Type	Message Length
Transaction ID	
Transaction ID	
Transaction ID	
Transaction ID	

每一个消息中带有 0 个或多个消息属性,它仅跟在消息头后,其中消息属性中包括一个 16 位的属性类型,一个 16 位的属性长度和可变的属性值。属性类型定义如下:

Type	Length
Value	

属性类型定义如下:

MAPPED-ADDRESS	0x0001
USERNAME	0x0006
MESSAGE-INTEGRITY	0x0008
ERROR-CODE	0x0009
UNKNOWN-ATTRIBUTES	0x000a
TRANSPORT-PREFERENCES	0x000c
LIFETIME	0x000d
BANDWIDTH	0x0010
ALTERNATE-SERVER	0x000e
MAGIC-COOKIE	0x000f

注意:在本文的设计中,MAGIC-COOKIE 属性没有任何作用,但为了考虑兼容性,TURN 消息结构中仍然定义了该属性。

1) MAPPED-ADDRESS 格式如下:

x x x x x x x x	Family	Port
Address		

MAPPED-ADDRESS 属性表示 TURN 服务器为其分配的 IP 地址和端口,它包括一个 8 位的地址家族(IPV4/IPV6),一个 16 位的端口和 32 位的固定长度的 IP 地址。

2) TRANSPORT-PREFERENCES 格式如下:

0	P	Type
x x x x x x x x	Family	Port
Address		

TRANSPORT-PREFERENCES 属性表示 TURN 服务器根据属性中 Type 值选择分配哪类端口(奇、偶或无要求)。具体设计如下:

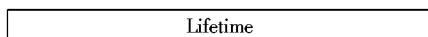
TYPE 类型:

- 0b00: 无要求
- 0b01: 想要服务器给它分配一个奇端口
- 0b10: 想要服务器给它分配一个偶端口

0b11: 请求一个已预分配的端口

P 位: 表示是否想要一个预分配端口。如果为 1, 表示想要哪类的预分配端口, 且长度为 32 位; 如果为 0, 则表示请求一个已经预分配的端口, 且长度变为 96 位, family, Port 和 Address 的格式和 MAPPED-ADDRESS 相同。

3) LIFETIME 的格式如下:

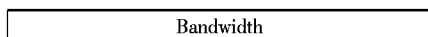


Lifetime 属性表示: 在绑定的会话中, 没有会话流量时的生命周期。如果 Lifetime 为 0, 则释放掉绑定的会话, 其他会话可以重新使用这些地址端口。它是一个 32 位的数据, 以秒为单位。

4) ALTERNATE-SERVER 的格式:

它和 MAPPED-ADDRESS 格式相同, 在该设计中主要用于 TURN 服务器的冗余。

5) BANDWIDTH 的格式如下:



BANDWIDTH 属性表示客户端期望使用的峰值速率 (kbit/s), 它是一个 32 位的整数。

6) MORE-AVAILABLE 的格式如下: 为了为客户机预分配多个传输地址, 在服务器端采用了 MORE-AVAILABLE 属性, 该属性的语法结构和 MAPPED-ADDRESS 相同, 表示服务器为客户机预分配的 IP 地址和端口。

TURN 消息的数据结构如下:

```
typedef struct
{
    TurnMsgHdr msgHdr;
    bool hasMappedAddress;
    TurnAttrAddress4 mappedAddress;
    bool hasUsername;
    TurnAttrString username;
    bool hasPassword;
    TurnAttrString password;
    bool hasMessageIntegrity;
    TurnAttrIntegrity messageIntegrity;
    bool hasErrorCode;
    TurnAttrError errorCode;
    bool hasUnknownAttributes;
    TurnAttrUnknown unknownAttributes;
    bool hasTransport_Preferences;
    TurnAttrPreferences transport_PreferencesAttributes;
    bool hasLifetime;
    TurnAttrInt32 lifetimeAttributes;
    bool hasMagic_Cookie;
    TurnAttrInt32 magic_CookieAttributes;
    bool hasBandWidth;
    TurnAttrInt32 bandwidthAttributes;
    bool hasAlternate_Server;
    TurnAttrAddress4 alternate_Server;
    bool hasMoreAliavable;
    TurnAttrMoreAliavable moreAliavableAttributes;
} TurnMessage;
```

4 TURN 服务器的实现

在该服务器设计中, 本文采用的是带外管理, 对于信令和数据是分开处理的。这种设计可以加快服务器的处理速度, 减少包的延迟, 因为服务器不再对 MAGIC-COOKIE 属性进行处理来判断是信令还是数据。另外本文对于地址映射表的设

计, 没有采用参考草案中的五原子映射表, 而是采用了一种简化的数据结构, 这种设计不仅精简了关键的数据结构, 还能简化处理, 因此, 在一定程度上减少了 TURN 服务器包的部分延迟。地址映射表的数据结构如下:

```
#define LIST_ENTRY( type )
struct {
    struct type * le_next;           /* 下一个成员 */
    struct type ** le_prev;         /* 上一个成员 */
}
struct session {
    LIST_ENTRY( session ) link;     /* 会话映射表的入口 */
    struct sockaddr addr1;          /* 发起请求的源地址 */
    struct sockaddr addr2;          /* 接受请求的目的地址 */
    int fd;                          /* IPv4 套接字 */
    int fd6;                         /* IPv6 套接字 */
    int port;                        /* 被映射的公网端口, IP 地址是 TURN 服务器的地址 */
    int cleanup_in;                 /* 超时计数器 */
    char * call_id;                 /* 会话的唯一标识符 */
};
```

4.1 信令处理

首先 TURN 服务器的控制套接字一直处于监听状态, 当它接收到消息后, 就将源的 IP 地址和端口 (因为这个地址是经 NAT 后的地址, 在某一段时间内是唯一的) 组合为会话中的 call ID (注意区别于 TURN 消息中的 call-ID), 这是一次会话的唯一标识, 然后判断 TURN 消息类型, 对于不同的消息类型进行不同的处理。

4.1.1 Shared Secret Request

本文在设计中, 对服务器和客户机采用了一种为客户机提供连接资源的机制, 在 TURN 服务器的数据库中为客户机提供了用户名和密码, 客户机要想和服务器连接, 必须知道自己在服务器上的权限。在该设计中, 本文采用 MD5 算法, 这样可以提供更高的安全性。

当服务器接收到一个 Shared Secret Request 请求时, 首先验证是否是 TLS 方式传输, 如果不是, 则产生一个 Shared Secret Error Response 响应 (包括一个 433 响应码的 ERROR-CODE 属性); 如果是 TLS 方式传输, 服务器检查 Shared Secret Request 请求消息的 MESSAGE-INTEGRITY 属性, 如果没有该属性, 服务器产生一个 420 响应码的 Shared Secret Error Response 响应 (包括 ERROR-CODE, 随机的 NONCE, REALM 等属性); 如果 MESSAGE-INTEGRITY 属性存在, 则服务器检查 REALM 域, 如果 REALM 属性不存在, 则产生一个 434 响应码的 Shared Secret Error Response 响应 (包括一个 NONCE 和 REALM 等属性值)。如果 REALM 域存在, 则服务器检查 NONCE 属性值, 如果不存在, 服务器产生一个 435 响应码的 Shared Secret Error Response 响应 (包括一个 NONCE 属性值和一个 REALM 属性值); 如果 NONCE 属性值存在, 则检查 USERNAME 属性值, 如果不存在, 服务器产生一个 432 响应码的 Shared Secret Error Response (包括一个 NONCE 属性值和一个 REALM 属性值); 如果 USERNAME 存在, 服务器计算 HMAC (MD5 (USERNAME-value, REALM-value, passwd, nonce), 将计算的 HMAC 和请求中的 HMAC 中的值比较, 如果相同, 说明用户名和密码正确, 则产生一个 Shared Secret Response; 否则, 产生一个 431 响应码的 Shared Secret Error Response (包括一个 NONCE 和 REALM 等属性值)。Shared Secret Response 响应包括服务器随机产生的一次性 USERNAME 和 PASSWORD 属性, 服务器将这两个属性保存

在会话权限表中。会话权限表包括用户名、密码、有效时间和会话的ID号等,服务器必须维护这个会话权限表。当某一个会话权限的有效时间到期后,将被自动释放,客户机必须重新申请权限。在后续请求中,客户机使用 USERNAME 和 PASSWORD 属性作为自己的权限向服务器发送分配请求。

如果服务器授权客户机使用服务器资源,则服务器为该客户机提供应用服务,为该客户机提供对外的公共传输地址(包括IP地址和端口),并产生一个 Shared Secret Response。

4.1.2 Allocate Request

客户机使用 Allocate requests(分配请求)获得公网的IP地址和端口,然后这个客户机就可以在网络上使用这个传输地址接收任何主机的数据。服务器为客户机分配一个本地的传输地址,并打开该地址,然后通过一个 Allocate Response(分配响应)消息通知客户机对外的传输地址。当服务器在该地址上接收了一个包后,扮演 TURN 中继角色,将该数据转发给发送分配请求的源地址。

因此,服务器必须维护一个会话绑定映射表。绑定的会话结构主要包括源地址、目的地址、绑定的套接字、已分配的端口、空闲超时时间、会话的ID号和数据包个数。

当服务器接收一个 Allocate Request 后,具体动作依赖于分配请求是初始化请求还是后续请求。初始化请求仅和源地址相关,和绑定地址映射表无关。后续请求不仅和源地址相关,还和绑定地址映射表有关。

1) Initial Requests

TURN 服务器必须准备接收 TCP/UDP 上的绑定请求,当 Allocate Request(分配请求)到达后,首先检查消息的完整性(MESSAGE-INTEGRITY)。如果完整性不存在,则服务器产生一个错误响应码为 401 的 Allocate Error Response。

如果消息完整性存在,则检查 USERNAME 属性;如果不存在,产生一个错误码为 432 的 Allocate Error Response。

如果 USERNAME 存在,则通过请求中提供的用户名和服务器上会话权限表中保存的密码计算 HMAC,比较 Allocate Request 中的 HMAC 和计算的 HMAC 是否相同,如果不相同,则产生一个错误码为 433 的 Allocate Error Response;如果相同,服务器检查其他属性。

如果有服务器不理解的属性,则产生一个错误码为 420 的 Allocate Error Response,并且这个错误响应包括一个 UNKNOWN-ATTRIBUTES 属性。

如果服务器已授权客户机请求,则服务器为该客户机分配传输地址(对外的IP地址和端口)。服务器首先查找分配请求中的 BANDWIDTH 属性,如果存在,则服务器将检查自己是否有足够的能力分配要求的带宽。

如果可以分配要求的带宽,则服务器查找请求中的 TRANSPORT-PREFERENCES 属性,如果属性要求分配一个偶数端口,则服务器为该客户机分配一个偶数端口;如果属性要求分配一个奇数端口,则为该客户机分配一个奇数端口;其他类似。

TRANSPORT-PREFERENCES 属性也可以表示对于一个端口的优先选择,服务器通过接收前一个分配请求,为其预分配一个端口,服务器通过分配响应中 MORE-AVAILABLE 属性通知客户机为其预分配的地址和端口。

注意:在该设计中,服务器一定不要为客户机分配公知的 0~1023 范围的端口,一定也不要分配用户注册的端口(1024~49151)。

如果满足带宽要求的端口不能被分配,则服务器产生一

个响应码为 300 的 Allocate Error Response。响应中可以包括一个 ALTERNATE-SERVER 属性(包括IP地址和端口),通过向该地址请求,客户机可以得到一个要求的端口。

如果一个端口被分配或预分配,服务器就在映射表中创建一个绑定的映射会话。在本文设计中,服务器总是为客户机预分配一个端口(RTCP协议使用)。如果一个地址和端口被预分配,则也启动一个定时器。如果在设置的时间内没有数据传输,则端口被释放,绑定的地址被删除,服务器可以重新使用。如果 LIFETIME 属性出现在请求中,但比服务器在绑定地址中使用的时间大,则服务器就降低 LIFETIME 值;服务器不增加请求中 LIFETIME 值。如果没有 LIFETIME 属性,则服务器就使用默认值。在任何一种情况下,服务器都将这个时间添加到绑定地址中,并启动定时器开始计时。

一旦端口或预分配端口被获得,则服务器启动定时器,并产生一个 Allocate Response。该响应包括一个 MAPPED-ADDRESS(IP地址和端口),LIFETIME(以秒为单位)、BANDWIDTH(如果请求中有 BANDWIDTH,则和请求中的相同)和 MESSAGE-INTEGRITY 属性。如果服务器按照自己的判断为客户机预分配一个传输地址(地址和端口),服务器在分配响应中必须包括一个 MORE-AVAILABLE 属性,这个属性包括预分配的传输地址(地址和端口)。

最后,服务器按原路发送分配响应给客户机。

2) Requests for Pre-Allocated Ports

TRANSPORT-PREFERENCES 属性表示前一个请求想要分配的端口,如果该属性出现,则服务器检查为前一个 Allocate Request 预分配的IP地址和端口。如果不同的协议到达,则发送一个错误码为 400 的 Allocate Error Response;如果被请求的端口已经为相同的用户预分配,则服务器将在地址映射表中绑定地址,并启动定时器,产生一个最终的 Allocate Response,这个响应包括一个 MAPPED-ADDRESS、LIFETIME 和 MESSAGE-INTEGRITY。

4.2 数据处理(Relay 转发)

当服务器的最终分配响应返回时,建立会话映射表,然后将该绑定的空闲端口打开并一直监听,等待与参加这次会话的任何一方通信。如果某一个套接字接收到数据,则判断会话的两个地址是否为空。如果一个不为空,并且源地址等于该会话中的某一个地址,则判断另一个地址是否为空,若不为空,则向另一个地址转发,并重新置位会话映射表中该会话的时间;若为空,则等待另一个地址被填充,并重新置位会话映射表中该会话的时间。如果两个都不为空,且源地址等于其中的一个地址,则转发;如果源地址不等于任何一个地址,则丢弃该数据报,继续等待接收。

在设置的一个固定时间内,如果会话映射表中的任何一次会话没有任何数据转发,则关闭该会话,并释放被分配的该端口,以供其他客户机重新使用。在服务器端,会话映射表是一个全局数据结构,超时成员受定时器任务控制,每到一秒,如果没有数据转发,则超时成员减1;如果减到0,则该会话就从会话映射表中自动被删除。

5 结语

在SIP服务器的研究课题中,本文以Linux操作系统为平台,实现了穿越NAT的一种技术——TURN服务器原型系统的设计和实现。经测试,功能完备,性能稳定。并且,NAT设备不需要解析报文,不会增加NAT设备的负担,可扩展性好。

(下转第1715页)

b) 设有最小群 $R_i = \{p_1, p_2, \dots, p_n\}$ 和 $R_j = \{q_1, q_2, \dots, q_m\}$, 我们定义: $s_{i,j}$ 为 R_i 中有链接指向 R_j 中某页的网页的数目, $d_{i,j}$ 为 R_j 中有链接指向 R_i 中某页的网页的数目, $|R_i|$ 是 R_i 中网页的数目, $|R_j|$ 是 R_j 中网页的数目。

c) 合并的条件是: $|R_i| = s_{i,j} = d_{i,j} = |R_j|$ 。

图2中的最小群a和最小群b, 它们都含有3个collection, 而且每个标记为a的网页都有一个超链接指向b。所以这两个最小群满足合并条件, 可以合并形成一个含有3个collection的群。同样道理可以把最小群c和最小群d也合并进去。合并结果为相似群。

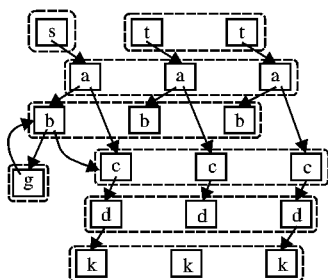


图2 最小群示例

1.5 各类方法比较

DSC, I-Match 等方法将网页作为单独的个体, 所有的网页存放在一个大的集合里, 两两进行比较, 以提高搜索结果的可用度。

网页集合比较方法将网页作为集合, 查找备份或是镜像, 若是备份或镜像, 则只用查找其中的一个即可, 节约了搜索时间。

总的来说: 若要面向大众化, 使用第一种方法会比较好一点。如像“球赛”等大众化的话题, 不可能会有服务器备份这种内容, 就算有, 它们的更新率很高, 可能过几天内容就已经完全不同了。第二种方法只有在查找较专业内容时比较好。

2 本系统使用的方法

考虑本系统的定位, 采集本专业相关的信息资料, 因此, 在参考网页及集合排重方法的基础上, 本系统使用了相似性检索技术进行自动排重。相似性检索是指对于给定样本文献, 在文献数据集中查找出与之内容相似的文献的技术。

相似性检索技术需要在文献数字化表示(空间向量模型VSM)的基础上, 通过计算文献之间的相似程度(向量之间的距离)给出文献之间的相关度指标。

本算法主要基于特征词提取和倒排索引技术:

- 1) 对样本库中的每篇文档进行自动分词和提取特征词;
- 2) 对样本库中的文档按特征词建立倒排索引库, 建立索引的相关属性, 包括词频、位置以及文本长度等;
- 3) 在进行相似性检索时, 对输入的检索文档分词并提取

特征词, 然后按特征词在倒排索引库中查找到与之相关的文档, 获得词频、位置及文本长度等属性;

4) 根据每篇文档中包含特征词的多少、位置、词频、文档的长度等信息来计算样本库中文档与待检索文档的相关度, 相关度超过一定阈值的文档即可作为相关文档处理, 并给出相关系数;

5) 系统为了提高排重的效果, 还增加了同义词库, 从语义上扩大相重信息的含义。

3 本系统的排重测试及评价

利用采集器从互联网上下载网页测试排重的效果。首先利用没有加入排重功能的采集器从互联网上下载10多万篇网页。分别利用前500篇、1000篇网页作为样本库, 然后采集器加入排重模块, 重新下载10多万篇网页, 测试结果如表1。

表1 测试结果

样本组成	召回率(R)	准确率(P)
前500篇作为样本库	80.0%	94.5%
前1000篇作为样本库	82.0%	90.5%

召回率(Recall) =

算法发现的重复网页数/网页总数 $\times 100\%$

准确率(Precision) =

正确重复的网页数/算法发现的重复网页数 $\times 100\%$

由于测试的网页较多, 无法一一判断哪些是重复的网页, 我们将算法发现的重复网页分为10份, 在这10份中随即挑选100个网页, 人工判断这100个网页的准确率, 用这10个准确率的平均值作为算法的准确率。结果表明, 采用此算法自动排重的平均准确度可以达到90%以上, 可以满足企业的实用化要求。

参考文献:

- [1] CHO J, SHIVAKUMAR N, GARCIA-MOLINA H. CA 94305, Finding replicated web collections[R]. Department of Computer Science Stanford, 1999.
- [2] 鲍军鹏, 沈钧毅, 刘晓东, 等. 自然语言文档复制检测研究综述[J]. 软件学报, 2003, 14(10).
- [3] CHOWDHURY A, FRIEDER O, GROSSMAN D, et al. Collection Statistics for Fast Duplicate Document Detection[J]. ACM Transactions on Information System, 2002, 20(2): 171-191.
- [4] LOPRESTI DP. Models and Algorithms for Duplicate Document Detection Bell Labs[A]. Proceedings of the Fifth International Conference on Document Analysis and Recognition[C], 1999.
- [5] CAMPBELL DM, CHEN WR, SMITH DM. Copy Detection Systems for Digital Documents[A]. Advances in Digital Libraries 2000 (ADL 2000)[C], 2000.

(上接第1691页)

因此, 在未来的应用中, 可以很好地满足穿越NAT, 在小区/企业网内部使多媒体通信变成现实。但是, 由于所有报文都必须经过TURN Server转发, 包的延迟和丢包的可能性仍然成为通信的瓶颈, 这是要进一步解决的问题。

参考文献:

- [1] ROSENBERG J, SCHULZRINNE H, CAMARILLO G, et al. RFC3261, SIP: Session Initiation Protocol[S], 2002.
- [2] HANDLEY M, JACOBSON V. RFC2327, SDP: Session Description Protocol[S], 1998.
- [3] SCHULZRINNE H, CASNER S, FREDERICK R, et al. RFC3550,

RTP: A Transport Protocol for Real-Time Applications[S], 2003.

- [4] ROSENBERG J, MAHY R, HUITEMA C. Traversal Using Relay NAT (TURN) [EB/OL]. draft-rosenberg-midcom-turn-02, 2003-10.
- [5] FRANKSJ, HALLAM-BAKERP, HOSTETLERJ, et al. RFC2617, HTTP Authentication: Basic and Digest Access Authentication[S], 1999.
- [6] HUITEMA C. RFC3605, Real Time Control Protocol (RTCP) attribute in Session Description Protocol (SDP)[S], 2003.
- [7] ROSENBERG J, WEINBERGER J, MAHY R. RFC3489, STUN - Simple Traversal of User Datagram Protocol (UDP) Through Network Address Translators (NATs)[S], 2003.