## DNS Server 资源记录类型

我们知道每个域需要指定主机来实现dns服务的功能，所以有类似以下字段：

baidu.com. NS ns.baidu.com. (指定哪个主机实现管理域名服务器功能)

ns.baidu.com. A 1.1.1.1 (这个主机在哪个位置)

实际上ns.baidu.com. A 1.1.1.1 这条记录在本机是可以不用指定的，因为真正授权是在上级指定的，也就是说以上两条记录应该在com这个域的域名数据库中指定，当然本机自身也要知道自己是谁，所以需要第一条记录，而既然本机可以成为ns服务器，说明本机的ip就是com指定的ip，所以第二条记录可以不要，之所以写上去是为了防止有人解析ns服务器的主机名A记录，此时我们需要返回结果。

dns是如此的重要，万一dns服务器挂掉，会造成很多域名无法解析。所以，为了防止一台单点故障导致dns服务无法使用，所以dns服务提供主从服务。当然dns也允许单机工作，根据它们功能的不通可以分为以下几种类型：

主域名服务器：主域名工作在主从模式下，在这个模式下dns可以在任意节点上解析，但是对dns的修改必须在主域名服务器上

从域名服务器：从服务器提供dns解析功能，它的数据来源于主域名服务器的拷贝，从服务器定时像主服务器请求更新数据文件。

缓存域名服务器（caching only server）：如果一个dns服务器本身不管理任何域，但是它可以把它收到的信息存储下来，并提供给别人解析，那么这就是一个缓存dns服务器

转发服务器（forwarding server）：如果一个dns服务器既不管理某个域也不解析任何dns，只为主机转发dns请求，那么这就是一个转发服务器。

DNS的主从工作原理：

dns主从服务器同时对外提供服务，所以要时刻保持数据的一致。为此，当主dns服务器数据库更新的时候从服务器必须到主服务器上cp数据到本地更新。但是从服务器怎么知道数据库改变了呢？它是这样子的：

主服务器和从服务器中都记录一个序列号(serial number),因为所有的更新必须在主dns服务器上操作，所以每次更新dns数据库的时候，主dns就在序列号上+1，这样子，当从服务器发来查看更新请求时，对比下序列号，就可以知道自己是否需要更新。

那么，从服务器什么时候来检查是否更新呢？

同样地，dns服务器记录一个刷新时间（refresh time），每隔这个时间从服务器来检查一次更新。然而更新过程中可能因为网络故障等问题无法联系上主dns服务器，这个时候，如果主dns刚好更新数据库，而我们又因为更新失败而等待下一次更新时间的话，那么dns缓存过期就会带来长时间的解析错误，这是我们不愿意看到的，所以dns服务器还定义了一个重试时间（retry time），定义当请求更新失败时，重新发起请求的时间间隔。所以一般这个时间比刷新时间小得多。但是，如果主dns服务器已经挂了，从服务器会一直请求更新失败，那么这个时候就需要一个标准来认定主dns挂了，这个就是过期时间(expire time)，当超过这个时间，从服务器会认为主服务器挂掉，然后从服务器自己也停止提供服务（这个是听说的额，我自己一直以为从服务器会替代主服务器，下一篇实战测试下）

所以一个dns服务器的数据库会包含以下信息：

序列号（serial number）

刷新时间（refresh time）

重试时间（retry time）

超时时间（expire time）

否定回答超时时间（negative answer ttl）

事实上，它还需定义否定回答的超时时间，这个时间可以防止别人恶意访问.

资源记录（resource record RR）：

我们把dns数据库中每一个条目称作一个资源记录（resource record RR）

我们知道，dns是用来做名称解析的，所以一个资源必须包含要解析的对象（name）和解析出来的结果(value)

我们还知道，以为缓存的关系，所以解析出来的结果需要一个过期时间，所以资源记录还需要ttl（time to live)值

有时我们需要把域名转化为ip，有时又相反，这是两种不同的过程，所以资源记录还需要类型（rrt resource record type）

所以一个资源记录格式大概如下：

NAME [ttl] IN RRT VALUE

※IN的意思是「Internet」，不是IN/OUT的「IN」

IN表示这是一个一个INTERNET的DNS资源记录，ttl加[]是因为，当所有的RR都有相同TTL值的时候，我们可以把TTL值写在最上面，变成如下形式：

TTL 600

NAME IN RRT VALUE

那么资源记录有什么类型呢？常见的如下：

1.A记录和AAAA记录，即域名-->ip，格式如下：

ns.baidu.com. 600 IN A 192.168.1.1

A是fqdn-->ipv4，AAAA是fqdn-->ipv6，一个A代表32位，对ipv6不太熟，这里不提供案例..

2.反向记录（PTR，pointer）即ip-->fqdn,格式如下：

1.1.168.192.in-addr.arpa. 600 IN PTR ns.baidu.com.

反向记录的写法比较特殊，它的name是ip反过来写并且以in-addr.arpa结尾

3.ns记录，我们知道ns是用来指定哪个主机担任某个域的解析的，所以它是fqdn-->fqdn，并且还需要知道这个主机的地址，所以它一般包含一个ns记录和ns主机的A记录，格式如下：

baidu.com. 600 IN NS ns.baidu.com

ns.baidu.com. 600 IN A 192.168.1.1

4.SOA（start of Authority）记录，既然我们掌管某个域，那么我们就应该拥有这个域的某些管理信息（域名，主从服务器之间的同步信息）所以这条记录是很重要的，必须放在区域文件第一条，它的格式如下：

ZONE\_NAME TTL IN SOA FQDN ADMIN\_MAILBOX (

serial number

refresh time

retrytime

expiretime

negative answer ttl

)

zone\_name表示区域名（注意不是域名），ttl表示这条记录的过期时间，fqdn表示主dns服务器主机，admin\_mailbox表示管理员的邮箱，剩下的参数前面已经讲过了，需要补充说的是，里面的时间单位可以写成M、H的形式，因为**@**在区域文件中有特殊含义（**代表区域名**，在bind的配置文件中定义），所以邮箱格式中的@必须转化为.

时间单位：M（分钟）、H（小时）、D（天）、W（周）默认是秒

邮箱格式：1@163.com--> 1.163.com

例子：

@ 600 IN SOA ns.baidu.com. 1.163.com. (

1

1H

2W

3D

600 )

也可以这么写：

baidu.com. 600 IN SOA ns.baidu.com. 1.163.com 1 1H 2W 3D 600

5.mx记录：我们知道，我们发邮件一般发送地址都是xx@xx.com的形式，但是我们主机怎么知道xx是谁呢？它是哪台主机上的用户呢?所以dns服务器还必须提供mx记录，它告诉外界，哪台主机提供了邮件服务，哪台主机可以帮你找到用户，它的格式如下：

ZONE\_NAME TTL IN MX PRI VALUE

PRI字段代表邮件服务器的优先级，值为0-99，数字越小优先级越高，这是因为可能提供多台邮件服务器的缘故

6.cname（canonical name）记录，也叫作别名记录，即一个域名的别名，fqdn-->fqdn的形式，例子：

www2.baidu.com. 50 IN CNAME www.baidu.com.

这里别名是www2.baidu.com. ,真正的域名是www.baidu.com.

### TXT记录

* 示例：ns1.exmaple.com. IN TXT "联系电话：XXXX"
* 解释：【domain】 IN TXT 【任意字符串】

##### 干什么用呢？

一般指某个主机名或域名的说明，或者联系方式，或者标注提醒等等。

### SPF记录

SPF记录是TXT记录的一个运用。后面的备注需要按照指定的格式才能有效。

* 示例：exmaple.com. IN TXT "v=spf1 ip4:198.51.100.1 ~all"
* 解释：【domain】 IN TXT 【送信侧邮件服务器确认规则】

##### 干什么用呢？

从发信侧服务器设定到DNS上的这条记录中，读取信息，判断发信侧是否合法。  
如果不符合规则，那么按照约定的规则处理掉。  
跟MX记录正好相反。  
MX：我是收件服务器，你找我时，请参考我设定到DNS服务器上的MX记录。  
SPF：我是发信服务器，你接受邮件时，请参考我设定到DNS服务器上SPF规则。如果不是我发的信，你可以删掉或者接收。

##### SPF记录规则

格式：  
版本 空格 定义 空格 定义 （空格 定义的循环）  
跟着例子看的话，比较好理解。  
example.com. IN SPF "v=spf1 ip4:192.0.2.1 -all"

* + v=spf1 是版本。只出现一次。
  + ip4:192.0.2.1 第一个定义
  + -all 第二个定义

定义的格式。

* + 种类  
    | all | ip4 | ip6 | a | mx | ptr | exists | include|
  + 前缀  
    "+" Pass（通过）  
    "-" Fail（拒绝）  
    "~" Soft Fail（软拒绝）  
    "?" Neutral（中立）

定义测试  
测试时，将从前往后依次测试每个定义。  
如果一个定义命中了要查询的 IP 地址，则由相应定义的前缀决定怎么处理。默认的前缀为+。  
如果测试完所有的 定义也没有命中，则结果为 Neutral。  
结果及处理方法一览

| **结果** | **说明** | **服务器处理办法** |
| --- | --- | --- |
| Pass | 发件IP是合法的 | 接受来信 |
| Fail | 发件 IP 是非法的 | 退信 |
| Soft Fail | 发件 IP 非法，但是不采取强硬措施 | 接受来信，但是做标记 |
| Neutral | SPF 记录中没有关于发件 IP 是否合法的信息 | 接受来信 |
| None | 服务器没有设定 SPF 记录 | 接受来信 |
| PermError | 发生了严重错误（例如 SPF 记录语法错误） | 没有规定 |
| TempError | 发生了临时错误（例如 DNS 查询失败） | 接受或拒绝 |