交换机 路由器

交换机表及表项

地址	接口	时间
62-FE-F7-11-89-A3	1	9:32
……	……	……

- **地址：** MAC地址
- **接口：** 通向该MAC地址的交换机接口
- **时间：** 表项的生存时间

交换机转发/过滤的工作流程

> e.g：从接口x收到目的地址为dd-dd-dd-dd-dd-dd的帧

- 表中无对应项：从除接口x外的所有接口转发出去
- 有目的地址为dd-dd-dd-dd-dd-dd的帧，但接口为x：丢弃
- 有目的地址为dd-dd-dd-dd-dd-dd的帧，且接口y≠x：从y转发出去

交换机自学习流程

对于收到的每个帧，交换机在表中存储：①该帧的源MAC地址；②该帧到达的接口；③到达时间

老化：若一段时间后未收到以该地址作为源地址的帧，则从表中删去该表项

拓扑限制，广播风暴

- 为了防止广播帧循环，交换网络的活跃拓扑得限制为一棵生成树

  交换机对于广播风暴不提供任何保护措施，如果某主机出了故障并传输没完没了的以太网广播帧流，则交换机会转发所有这些帧，使整个以太网崩溃

  参考文章（关键字：生成树协议）： 如何理解STP生成树协议？ - 车小胖的回答 - 知乎 https://www.zhihu.com/question/21327750/answer/109458339

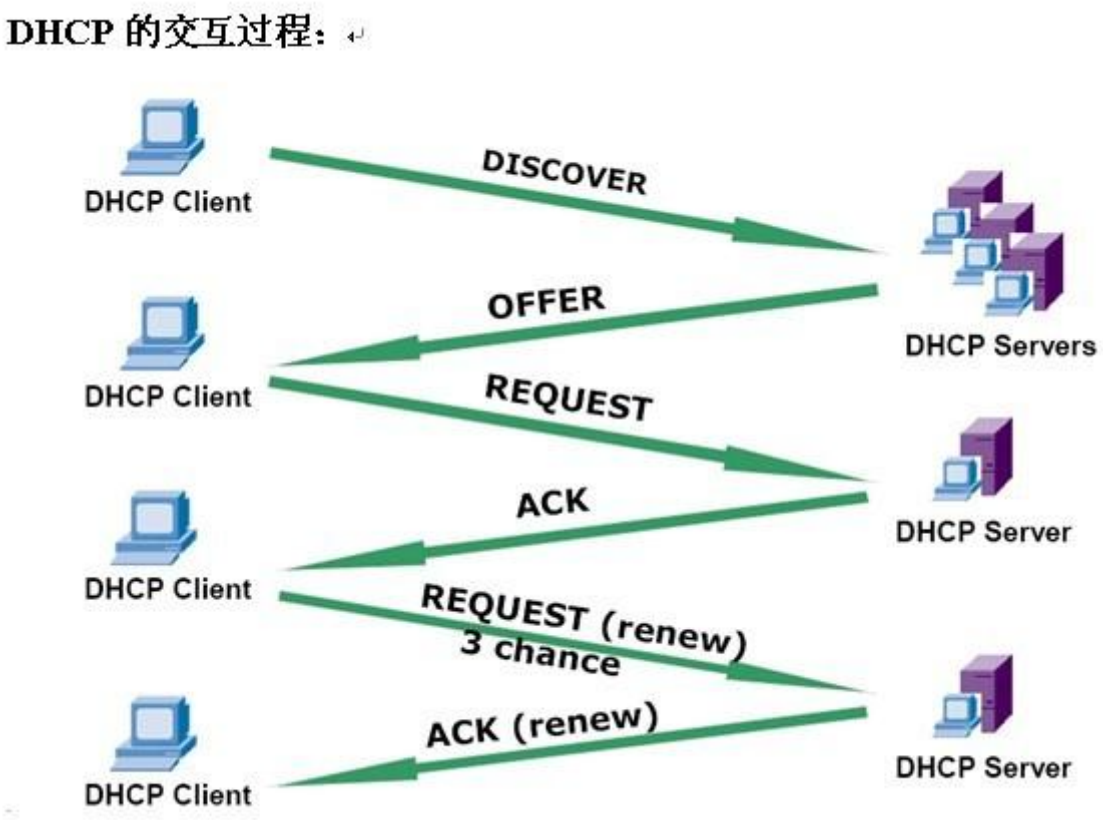
- 路由器没有拓扑限制，因为IP首部有TTL限制，报文转着转着就没了

  参考文章： https://zhidao.baidu.com/question/2475052.html

  路由器隔离广播风暴的原理：其实很简单，路由器不会将某个接口收到的广播包转发到另外一个接口所在的网络，因此，一个网段内的广播风暴不会对另一个网段造成任何影响，但是划分子网可就未必了，因为如果只是在同一个物理网络中划分的子网：比如192.168.0.0/24和192.168.1.0/24这两个网段中间并没有路由器的分割都是直接通过交换机相连的话，一旦某一台主机大量发送广播包，所有网段主机的数据通讯都会受到影响，和在一个网段的情形一样。

DHCP

DHCP交互过程如下图，不做详细介绍了



这个图是RFC文档中关于DHCP报文的介绍

```
Message         Use
-------         ---

DHCPDISCOVER -  Client broadcast to locate available servers.

DHCPOFFER    -  Server to client in response to DHCPDISCOVER with
                offer of configuration parameters.

DHCPREQUEST  -  Client message to servers either (a) requesting
                offered parameters from one server and implicitly
                declining offers from all others, (b) confirming
                correctness of previously allocated address after,
                e.g., system reboot, or (c) extending the lease on a
                particular network address.

DHCPACK      -  Server to client with configuration parameters,
                including committed network address.
```

为什么 DHCP REQUEST 仍然采用广播而非单播?

注意到 `requesting offered parameters from one server and implicitly declining offers from all others`，翻译一下就是：选择一个DHCP服务器并向它请求所提供的的参数（这些参数是 `DHCP OFFER` 报文里的），并且 显式的拒绝其他DHCP服务器

再附上百度百科中的说法：

- DHCP Server收到DHCP Request报文后，判断选项字段中的IP地址是否与自己的地址相同。 如果不相同，DHCP Server不做任何处理只清除相应IP地址分配记录；如果相同，DHCP Server就会向DHCP Client响应一个DHCP ACK报文，并在选项字段中增加IP地址的使用租期信息

- 正式请求DHCP服务器分配地址DHCPREQUEST采用广播包，是为了让其它所有发送DHCPOFFER数据包的DHCP服务器也能够接收到该数据包，然后释放已经OFFER（预分配）给客户端的IP地址

DHCP OFFER真的只能广播吗?

> If the 'giaddr' field in a DHCP message from a client is non-zero, the server sends any return messages to the 'DHCP server' port on the BOOTP relay agent whose address appears in 'giaddr'. If the 'giaddr' field is zero and the 'ciaddr' field is nonzero, then the server unicasts DHCPOFFER and DHCPACK messages to the address in 'ciaddr'. If 'giaddr' is zero and 'ciaddr' is zero, and the broadcast bit is set, then the server broadcasts DHCPOFFER and DHCPACK messages to 0xffffffff. If the broadcast bit is not set and 'giaddr' is zero and 'ciaddr' is zero, then the server unicasts DHCPOFFER and DHCPACK messages to the client's hardware address and 'yiaddr' address. In all cases, when 'giaddr' is zero, the server broadcasts any DHCPNAK messages to 0xffffffff.

首先科普一下DHCP报文段中的一些字段：

- **giaddr（gateway ip address）：** DHCP中继代理的IP地址
- **ciaddr（client ip address）：** 客户机IP地址
- **yiaddr（your ip address）：** 服务器给客户机分配的IP地址
- **siaddr（server ip address）：** 服务器的IP地址
- **chaddr（client hardware address）：** 客户机的硬件地址
- **bootp flag：** 引导协议（我也不知道干嘛的.....）

再看一下RFC文档中的描述：

- > If the 'giaddr' field is zero and the 'ciaddr' field is nonzero, then the server `unicasts` DHCPOFFER and DHCPACK messages to the address in 'ciaddr'.

  翻译：如果`giaddr`为0，但`ciaddr`不为0，则服务器向`ciaddr中的地址` **单播** DHCPOFFER和DHCPACK报文

  解读：没有DHCP中介且客户机已有IP地址则单播

- > If 'giaddr' is zero and 'ciaddr' is zero, and the broadcast bit is set, then the server `broadcasts` DHCPOFFER and DHCPACK messages to 0xffffffff.

  翻译：如果`giaddr`和`ciaddr`都为0，并且`设置了广播位`（broadcast bit），则服务器 **广播** DHCPOFFER和DHCPACK报文

  解读：正常教材介绍的多播情况

- > If the broadcast bit is not set and 'giaddr' is zero and 'ciaddr' is zero, then the server `unicasts` DHCPOFFER and DHCPACK messages to the `client's hardware address and 'yiaddr' address.`

  翻译：如果`giaddr`和`ciaddr`都为0，但`未设置了广播位`（broadcast bit），则服务器 **单播** DHCPOFFER和DHCPACK报文

  解读：由于DHCP协议字段中有`chaddr(硬件地址)`，而IP地址肯定是DHCP服务器分配的，DHCP有了客户端的IP地址、MAC地址，就能完成整个IP报文的封装从而实现单播了

- > In all cases, when 'giaddr' is zero, the server `broadcasts` any DHCPNAK messages to 0xffffffff.

  翻译：无论什么情况，只要`giaddr`为0，则服务器广播DHCPNAK报文

具体到操作系统好像没上面说的那么复杂.....（broadcast bit是boot flags字段中的1位）

- 对于`Linux系统`来说，DHCP的`bootp flags=0x0000`，发送的DHCP OFFER报文为单播
- 对于`Windows系统`来说，DHCP的`bootp flags=0x8000`，发送的DHCP OFFER报文为多播

如何设计一个数据库

无标准答案，仅给出一些参考文章以及个人思考角度

参考文章： https://www.cnblogs.com/457248499-qq-com/p/7382484.html

图片选自《高性能MySQL》

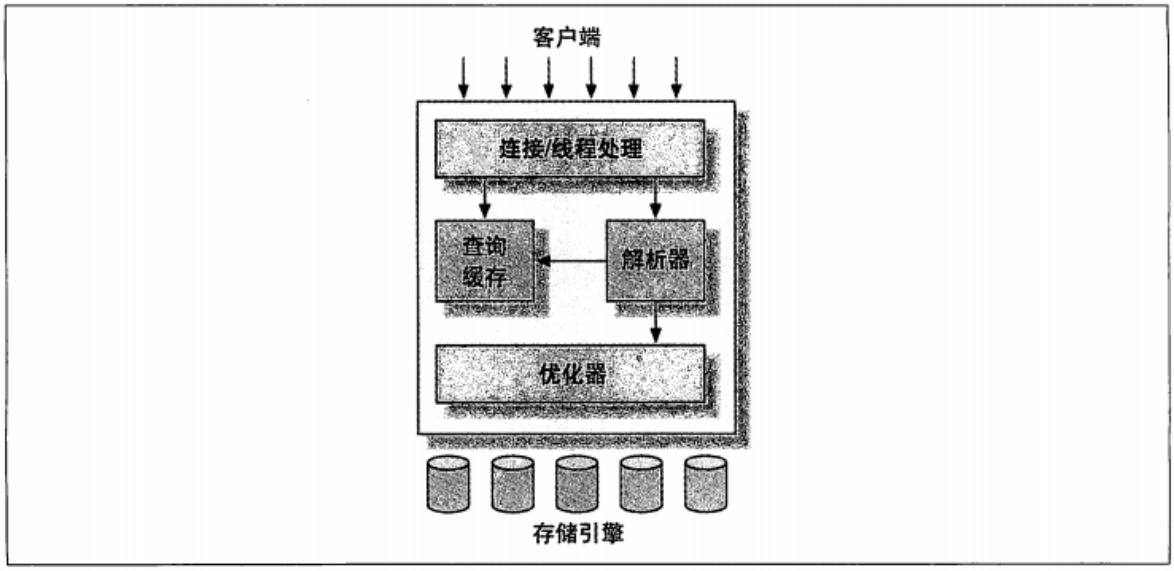


图1-1：MySQL服务器逻辑架构图

- **存储管理**：表的结构如何设计，如何存储一张表？怎么压缩只读的表以减少磁盘I/O次数？.....

```
mysql> show table status like 'orders' \G;
*************************** 1. row ***************************
           Name: orders
         Engine: InnoDB
        Version: 10
     Row_format: Dynamic
           Rows: 5
 Avg_row_length: 3276
    Data_length: 16384
Max_data_length: 0
   Index_length: 16384
      Data_free: 0
 Auto_increment: 20010
    Create_time: 2020-06-18 15:07:01
    Update_time: NULL
     Check_time: NULL
      Collation: latin1_swedish_ci
       Checksum: NULL
 Create_options:
        Comment:
```

- **SQL解析：**（涉及到编译原理了，我也不懂）

  MySQL 会解析查询，并创建内部数据结构（解析树），然后对其进行各种优化，包括重写查询、决定表的读取顺序，以及选择合适的索引等。用户可以通过特殊的关键字提示（hint）优化器，影响它的决策过程。也可以请求优化器解释（explain）优化过程的各个因素，使用户可以知道服务器是如何进行优化决策的，并提供一个参考基准，便于用户重构查询和 schema、修改相关配置，使应用尽可能高效运行。第 6 章我们将讨论更多优化器的细节。

  优化器并不关心表使用的是什么存储引擎，但存储引擎对于优化查询是有影响的。优化器会请求存储引擎提供容量或某个具体操作的开销信息，以及表数据的统计信息等。例如，某些存储引擎的某种索引，可能对一些特定的查询有优化。关于索引与 schema 的优化，请参见第 4 章和第 5 章。

- **缓存：**重复查询用的SQL语句及结果是不是应该缓存一下（具体缓存方法可能LRU、LFU...）

  对于 SELECT 语句，在解析查询之前，服务器会先检查查询缓存（Query Cache），如果能够在其中找到对应的查询，服务器就不必再执行查询解析、优化和执行的整个过程，而是直接返回查询缓存中的结果集。第 7 章详细讨论了相关内容。

- **持久化，日志管理，容灾机制：**

  事务日志可以帮助提高事务的效率。使用事务日志，存储引擎在修改表的数据时只需要修改其内存拷贝，再把该修改行为记录到持久在硬盘上的事务日志中，而不用每次都将修改的数据本身持久到磁盘。事务日志采用的是追加的方式，因此写日志的操作是磁盘上一小块区域内的顺序 I/O，而不像随机 I/O 需要在磁盘的多个地方移动磁头，所以采用事务日志的方式相对来说要快得多。事务日志持久以后，内存中被修改的数据在后台可以慢慢地刷回到磁盘。目前大多数存储引擎都是这样实现的，我们通常称之为预写式日志（Write-Ahead Logging），修改数据需要写两次磁盘。

  如果数据的修改已经记录到事务日志并持久化，但数据本身还没有写回磁盘，此时系统崩溃，存储引擎在重启时能够自动恢复这部分修改的数据。具体的恢复方式则视存储引擎而定。

- **权限划分：**防止程序员跑路的时候把库删了

- **索引管理：**如何加快SQL的查询效率，可以扯一下B+树索引、Hash索引

- **锁管理：**针对多个事务并发执行可能产生的问题，考虑的角度：锁的类型（读锁/写锁）、加锁策略（意向锁）、锁的粒度（表锁/行锁）、两段锁协议等等