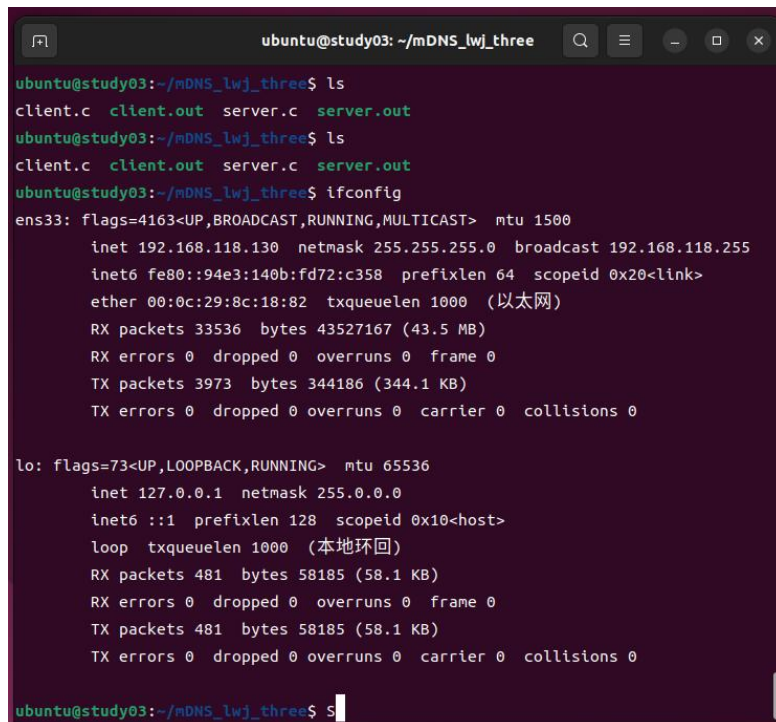


## 1.1 简介

利用两台虚拟机模拟两台局域网设备，第一台模拟器主机名 study03 运行服务端，第二台模拟器主机名 study02 运行客户端。客户端输出服务端主机名（study03.local）、IPv4（192.168.118.130）、IPv6 地址（fe80::94e3:140b:fd72:c358）、服务器名称（MQTTServer）、类型（\_mqtt.\_tcp）、地址（mqtts://tb.com）。

## 1.2 配置环境

### ①确保虚拟机在同一局域网



```
ubuntu@study03: ~/mDNS_lwj_three
ubuntu@study03:~/mDNS_lwj_three$ ls
client.c  client.out  server.c  server.out
ubuntu@study03:~/mDNS_lwj_three$ ls
client.c  client.out  server.c  server.out
ubuntu@study03:~/mDNS_lwj_three$ ifconfig
ens33: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500
    inet 192.168.118.130  netmask 255.255.255.0  broadcast 192.168.118.255
    inet6 fe80::94e3:140b:fd72:c358  prefixlen 64  scopeid 0x20<link>
    ether 00:0c:29:8c:18:82  txqueuelen 1000  (以太网)
    RX packets 33536  bytes 43527167 (43.5 MB)
    RX errors 0  dropped 0  overruns 0  frame 0
    TX packets 3973  bytes 344186 (344.1 KB)
    TX errors 0  dropped 0  overruns 0  carrier 0  collisions 0

lo: flags=73<UP,LOOPBACK,RUNNING>  mtu 65536
    inet 127.0.0.1  netmask 255.0.0.0
    inet6 ::1  prefixlen 128  scopeid 0x10<host>
    loop txqueuelen 1000  (本地环回)
    RX packets 481  bytes 58185 (58.1 KB)
    RX errors 0  dropped 0  overruns 0  frame 0
    TX packets 481  bytes 58185 (58.1 KB)
    TX errors 0  dropped 0  overruns 0  carrier 0  collisions 0

ubuntu@study03:~/mDNS_lwj_three$ S
```

```
user@study02: ~/Test
client.c client.out server.c server.out
user@study02:~/Test$ ./server.out
Service started. Press Ctrl+C to exit.
^C
user@study02:~/Test$ ifconfig
ens33: flags=4163<UP,BROADCAST,RUNNING,MULTICAST> mtu 1500
    inet 192.168.118.129 netmask 255.255.255.0 broadcast 192.168.118.255
    inet6 fe80::9631:17e7:e7fe:a754 prefixlen 64 scopeid 0x20<link>
    ether 00:0c:29:10:12:66 txqueuelen 1000 (以太网)
    RX packets 6116 bytes 2261962 (2.2 MB)
    RX errors 0 dropped 0 overruns 0 frame 0
    TX packets 1293 bytes 117956 (117.9 KB)
    TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0

lo: flags=73<UP,LOOPBACK,RUNNING> mtu 65536
    inet 127.0.0.1 netmask 255.0.0.0
    inet6 ::1 prefixlen 128 scopeid 0x10<host>
    loop txqueuelen 1000 (本地环回)
    RX packets 206 bytes 18623 (18.6 KB)
    RX errors 0 dropped 0 overruns 0 frame 0
    TX packets 206 bytes 18623 (18.6 KB)
    TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0

user@study02:~/Test$
```

根据 IP 地址可以确定两台虚拟机在同一个子网。

## ②确认虚拟机的网络模式



两台虚拟机均为 NAT 模式可以通信。

## ③确保防火墙没有阻止 UDP 5353 端口

```
ubuntu@study03:~/mDNS_lwj_three$ sudo ufw status verbose
[sudo] ubuntu 的密码:
状态: 不活动
ubuntu@study03:~/mDNS_lwj_three$
```

```
user@study02:~/Test$ sudo ufw status verbose
[sudo] user 的密码:
状态: 不活动
user@study02:~/Test$
```

两台虚拟机均关闭防火墙，如果开启可以用命令 `sudo ufw allow 5353/udp` 设置允许 5353 端口。

### 1.3 运行结果

开启 avahi 守护进程，检查状态（是否开启），第一台虚拟机运行服务端。

```
ubuntu@study03: ~/mDNS_lwj_three
Tasks: 2 (limit: 4551)
Memory: 844.0K
CPU: 160ms
CGroup: /system.slice/avahi-daemon.service
├─4548 "avahi-daemon: running [study03.local]"
└─4549 "avahi-daemon: chroot helper"

8月 18 23:17:01 study03 avahi-daemon[4548]: Joining mDNS multicast group on interface lo.IPv6 for >
8月 18 23:17:01 study03 avahi-daemon[4548]: New relevant interface lo.IPv6 for >
8月 18 23:17:01 study03 avahi-daemon[4548]: Joining mDNS multicast group on interface lo.IPv4 for >
8月 18 23:17:01 study03 avahi-daemon[4548]: New relevant interface lo.IPv4 for >
8月 18 23:17:01 study03 avahi-daemon[4548]: Network interface enumeration complete >
8月 18 23:17:01 study03 avahi-daemon[4548]: Registering new address record for >
8月 18 23:17:01 study03 avahi-daemon[4548]: Registering new address record for >
8月 18 23:17:01 study03 avahi-daemon[4548]: Registering new address record for >
8月 18 23:17:01 study03 avahi-daemon[4548]: Registering new address record for >
8月 18 23:17:02 study03 avahi-daemon[4548]: Server startup complete. Host name >

ubuntu@study03:~/mDNS_lwj_three$ sudo systemctl start avahi-daemon
ubuntu@study03:~/mDNS_lwj_three$ ls
client.c  client.out  server.c  server.out
ubuntu@study03:~/mDNS_lwj_three$ ./server.out
Service started. Press Ctrl+C to exit.
```

开启 avahi 守护进程，检查状态，第二台虚拟机运行客户端，输出服务端信息。

```
user@study02: ~/Test
lo: flags=73<UP,LOOPBACK,RUNNING> mtu 65536
    inet 127.0.0.1 netmask 255.0.0.0
    inet6 ::1 prefixlen 128 scopeid 0x10<host>
    loop txqueuelen 1000 (本地环回)
    RX packets 206 bytes 18623 (18.6 KB)
    RX errors 0 dropped 0 overruns 0 frame 0
    TX packets 206 bytes 18623 (18.6 KB)
    TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0

user@study02:~/Test$ sudo ufw status verbose
[sudo] user 的密码:
状态: 不活动
user@study02:~/Test$ ls
client.c  client.out  server.c  server.out
user@study02:~/Test$ sudo systemctl start avahi-daemon
user@study02:~/Test$ ./client.out
Service browser started.
Service Hostname: study03.local
Resolved IPv4 IP Address: 192.168.118.130
Resolved IPv6 IP Address: fe80::94e3:140b:fd72:c358
Service Name: MQTTServer
Service Type: _mqtt._tcp
Service Address (from TXT record): mqttts://tb.com
```

## 1.4 程序

### //服务端

```
1.  #include <avahi-client/client.h>
2.  #include <avahi-client/publish.h>
3.  #include <avahi-common/thread-watch.h>
4.  #include <avahi-common/error.h>
5.  #include <stdio.h>
6.  #include <stdlib.h>
7.  #include <string.h>
8.  #include <unistd.h>
9.  #include <ifaddrs.h>
10. #include <arpa/inet.h>
11.
12. int main() {
13.     // 创建 AvahiThreadedPoll 实例
14.     AvahiThreadedPoll *poll = avahi_threaded_poll_new();
15.     if(!poll) {
16.         fprintf(stderr, "Failed to create threaded poll.\n");
17.         return EXIT_FAILURE;
18.     }
19.
20.     // 初始化 Avahi 客户端
21.     AvahiClient *client = avahi_client_new(avahi_threaded_poll_get(poll), AVAHI_CLIENT_NO_FAIL, NULL, NULL, NULL);
22.
23.     if(!client) {
24.         fprintf(stderr, "Failed to create Avahi client.\n");
25.         avahi_threaded_poll_free(poll); // 释放线程池
26.         return EXIT_FAILURE;
27.     }
28.
29.     // 创建服务
30.     AvahiEntryGroup *group = avahi_entry_group_new(client, NULL, NULL);
31.
32.     // 修改服务名称和类型为 mqtt
33.     const char *service_name = "MQTTServer";
34.     const char *service_type = "_mqtt._tcp";
35.     const char *mqtt_server_address = "mqtt://tb.com";
36.
37.     // 获取本机的 IP 地址
38.     struct ifaddrs *ifaddr, *ifa;
39.     char ipv4_addr[INET_ADDRSTRLEN] = "";
40.     char ipv6_addr[INET6_ADDRSTRLEN] = "";
```

```
41.
42.     if(getifaddrs(&ifaddr) == -1) {
43.         perror("getifaddrs");
44.         return EXIT_FAILURE;
45.     }
46.
47.     for(ifa = ifaddr; ifa != NULL; ifa = ifa->ifa_next) {
48.         if(ifa->ifa_addr == NULL) continue;
49.
50.         int family = ifa->ifa_addr->sa_family;
51.         if(family == AF_INET) {
52.             inet_ntop(AF_INET, &((struct sockaddr_in *)ifa->ifa_addr)->sin_addr, ipv4_addr, INET_A
DDRSTRLEN);
53.         }
54.         else if(family == AF_INET6) {
55.             inet_ntop(AF_INET6, &((struct sockaddr_in6 *)ifa->ifa_addr)->sin6_addr, ipv6_addr, INE
T6_ADDRSTRLEN);
56.         }
57.     }
58.     freeifaddrs(ifaddr);
59.
60.     // 创建 TXT 记录
61.     AvahiStringList *txt = NULL;
62.     txt = avahi_string_list_add_pair(txt, "mqtt_server_address", mqtt_server_address);
63.     if(strlen(ipv4_addr) > 0) txt = avahi_string_list_add_pair(txt, "ipv4", ipv4_addr);
64.     if(strlen(ipv6_addr) > 0) txt = avahi_string_list_add_pair(txt, "ipv6", ipv6_addr);
65.
66.     if(avahi_entry_group_add_service_strlst(group, AVAHI_IF_UNSPEC, AVAHI_PROTO_UNSPEC, 0, service
_name, service_type, NULL, NULL, 0, txt) < 0) {
67.         fprintf(stderr, "Failed to add service TXT record: %s\n", avahi_strerror(avahi_client_errn
o(client)));
68.         avahi_client_free(client);
69.         avahi_threaded_poll_free(poll);
70.         return EXIT_FAILURE;
71.     }
72.
73.     // 提交服务
74.     if(avahi_entry_group_commit(group) < 0) {
75.         fprintf(stderr, "Failed to commit entry group: %s\n", avahi_strerror(avahi_client_errno(c
lient)));
76.         avahi_client_free(client);
77.         avahi_threaded_poll_free(poll); // 释放线程池
78.         return EXIT_FAILURE;
79.     }
```

```

80.
81.     // 启动事件循环
82.     avahi_threaded_poll_start(poll);
83.
84.     // 让程序持续运行
85.     printf("Service started. Press Ctrl+C to exit.\n");
86.     while (1) {
87.         pause(); // 等待信号
88.     }
89.
90.     // 释放资源
91.     avahi_entry_group_free(group);
92.     avahi_client_free(client);
93.     avahi_threaded_poll_free(poll);
94.
95.     return EXIT_SUCCESS;
96. }

```

## //客户端

```

1.  #include <avahi-client/client.h>
2.  #include <avahi-client/lookup.h>
3.  #include <avahi-common/thread-watch.h>
4.  #include <avahi-common/error.h>
5.  #include <avahi-common/malloc.h>
6.  #include <stdio.h>
7.  #include <stdlib.h>
8.  #include <unistd.h>
9.  #include <string.h>
10. #include <netinet/in.h>
11. #include <arpa/inet.h>
12.
13. // 标志位, 用于确保只打印一次服务信息
14. int service_printed = 0;
15.
16. int ipv4_printed = 0; // IPv4 解析完成标志
17. int ipv6_printed = 0; // IPv6 解析完成标志
18.
19. void resolve_callback(AvahiServiceResolver *resolver, AvahiIfIndex interface, AvahiProtocol protocol, AvahiResolverEvent event,
20.     const char *name, const char *type, const char *domain, const char *hostname,
21.     const AvahiAddress *address, uint16_t port, AvahiStringList *txt, AvahiLookupResultFlags flags,
22.     void *userdata) {
23.     if(event == AVAHI_RESOLVER_FOUND) {
24.         // 打印服务信息
25.         printf("Service found: %s %s %s %s\n", name, type, domain, hostname);
26.         if(address) {
27.             printf("Address: %s\n", avahi_address_string(address));
28.         }
29.         if(port) {
30.             printf("Port: %d\n", port);
31.         }
32.         if(txt) {
33.             printf("TXT: %s\n", avahi_string_list_to_string(txt));
34.         }
35.         if(flags & AVAHI_LOOKUP_RESULT_FLAG_TXT) {
36.             printf("TXT: %s\n", avahi_string_list_to_string(txt));
37.         }
38.         if(flags & AVAHI_LOOKUP_RESULT_FLAG_FLAGS) {
39.             printf("Flags: %d\n", flags);
40.         }
41.         if(flags & AVAHI_LOOKUP_RESULT_FLAG_FLAGS) {
42.             printf("Flags: %d\n", flags);
43.         }
44.         if(flags & AVAHI_LOOKUP_RESULT_FLAG_FLAGS) {
45.             printf("Flags: %d\n", flags);
46.         }
47.         if(flags & AVAHI_LOOKUP_RESULT_FLAG_FLAGS) {
48.             printf("Flags: %d\n", flags);
49.         }
50.         if(flags & AVAHI_LOOKUP_RESULT_FLAG_FLAGS) {
51.             printf("Flags: %d\n", flags);
52.         }
53.         if(flags & AVAHI_LOOKUP_RESULT_FLAG_FLAGS) {
54.             printf("Flags: %d\n", flags);
55.         }
56.         if(flags & AVAHI_LOOKUP_RESULT_FLAG_FLAGS) {
57.             printf("Flags: %d\n", flags);
58.         }
59.         if(flags & AVAHI_LOOKUP_RESULT_FLAG_FLAGS) {
60.             printf("Flags: %d\n", flags);
61.         }
62.         if(flags & AVAHI_LOOKUP_RESULT_FLAG_FLAGS) {
63.             printf("Flags: %d\n", flags);
64.         }
65.         if(flags & AVAHI_LOOKUP_RESULT_FLAG_FLAGS) {
66.             printf("Flags: %d\n", flags);
67.         }
68.         if(flags & AVAHI_LOOKUP_RESULT_FLAG_FLAGS) {
69.             printf("Flags: %d\n", flags);
70.         }
71.         if(flags & AVAHI_LOOKUP_RESULT_FLAG_FLAGS) {
72.             printf("Flags: %d\n", flags);
73.         }
74.         if(flags & AVAHI_LOOKUP_RESULT_FLAG_FLAGS) {
75.             printf("Flags: %d\n", flags);
76.         }
77.         if(flags & AVAHI_LOOKUP_RESULT_FLAG_FLAGS) {
78.             printf("Flags: %d\n", flags);
79.         }
80.         if(flags & AVAHI_LOOKUP_RESULT_FLAG_FLAGS) {
81.             printf("Flags: %d\n", flags);
82.         }
83.         if(flags & AVAHI_LOOKUP_RESULT_FLAG_FLAGS) {
84.             printf("Flags: %d\n", flags);
85.         }
86.         if(flags & AVAHI_LOOKUP_RESULT_FLAG_FLAGS) {
87.             printf("Flags: %d\n", flags);
88.         }
89.         if(flags & AVAHI_LOOKUP_RESULT_FLAG_FLAGS) {
90.             printf("Flags: %d\n", flags);
91.         }
92.         if(flags & AVAHI_LOOKUP_RESULT_FLAG_FLAGS) {
93.             printf("Flags: %d\n", flags);
94.         }
95.         if(flags & AVAHI_LOOKUP_RESULT_FLAG_FLAGS) {
96.             printf("Flags: %d\n", flags);
97.         }
98.         if(flags & AVAHI_LOOKUP_RESULT_FLAG_FLAGS) {
99.             printf("Flags: %d\n", flags);
100.        }
101.    }
102. }

```

```

23.     char address_str[AVAHI_ADDRESS_STR_MAX];
24.     avahi_address_snprint(address_str, sizeof(address_str), address);
25.
26.     if(protocol == AVAHI_PROTO_INET && !ipv4_printed) {
27.         printf("Service Hostname: %s\n", hostname);
28.         printf("Resolved IPv4 IP Address: %s\n", address_str);
29.         ipv4_printed = 1;
30.     }
31.     else if(protocol == AVAHI_PROTO_INET6 && !ipv6_printed) {
32.         printf("Resolved IPv6 IP Address: %s\n", address_str);
33.         ipv6_printed = 1;
34.     }
35.
36.     // 只有在两种地址都打印后, 才打印服务信息并设置标志位
37.     if(ipv4_printed && ipv6_printed && !service_printed) {
38.         printf("Service Name: %s\n", name);
39.         printf("Service Type: %s\n", type);
40.
41.         while(txt) {
42.             char *key, *value;
43.             avahi_string_list_get_pair(txt, &key, &value, NULL);
44.             if(strcmp(key, "mqtt_server_address") == 0) {
45.                 printf("Service Address (from TXT record): %s\n", value);
46.             }
47.             avahi_free(key);
48.             avahi_free(value);
49.             txt = avahi_string_list_get_next(txt);
50.         }
51.         service_printed = 1;
52.     }
53. }
54.
55.     avahi_service_resolver_free(resolver);
56. }
57.
58. void browse_callback(AvahiServiceBrowser *browser, AvahiIfIndex interface, AvahiProtocol protocol,
    AvahiBrowserEvent event,
59.     const char *name, const char *type, const char *domain, AvahiLookupResultFlags flags, void *us
    erdata) {
60.     AvahiClient *client = userdata;
61.     if(event == AVAHI_BROWSER_NEW) {
62.         avahi_service_resolver_new(client, interface, AVAHI_PROTO_INET, name, type, domain, AVAHI_
            PROTO_UNSPEC, 0, resolve_callback, client);

```

```

63.         avahi_service_resolver_new(client, interface, AVAHI_PROTO_INET6, name, type, domain, AVAHI
           _PROTO_UNSPEC, 0, resolve_callback, client);
64.     }
65. }
66.
67.
68. int main() {
69.     AvahiThreadedPoll *poll = avahi_threaded_poll_new();
70.     if(!poll) {
71.         fprintf(stderr, "Failed to create threaded poll.\n");
72.         return EXIT_FAILURE;
73.     }
74.
75.     AvahiClient *client = avahi_client_new(avahi_threaded_poll_get(poll), AVAHI_CLIENT_NO_FAIL, NU
           LL, NULL, NULL);
76.     if(!client) {
77.         fprintf(stderr, "Failed to create Avahi client.\n");
78.         avahi_threaded_poll_free(poll);
79.         return EXIT_FAILURE;
80.     }
81.
82.     AvahiServiceBrowser *browser = avahi_service_browser_new(client, AVAHI_IF_UNSPEC, AVAHI_PROTO_
           UNSPEC, "_mqtt.tcp", NULL, 0, browse_callback, client);
83.     if(!browser) {
84.         fprintf(stderr, "Failed to create service browser.\n");
85.         avahi_client_free(client);
86.         avahi_threaded_poll_free(poll);
87.         return EXIT_FAILURE;
88.     }
89.
90.     printf("Service browser started.\n");
91.
92.     avahi_threaded_poll_start(poll);
93.
94.     while(1) {
95.         pause();
96.     }
97.
98.     avahi_service_browser_free(browser);
99.     avahi_client_free(client);
100.    avahi_threaded_poll_free(poll);
101.
102.    return EXIT_SUCCESS;
103. }

```