Page : ( ) Year: Month: Day:

ادامهٔ ۷ب)

حالا باید ببینیم اندازهٔ ساختار (برحسب بایت) که sizeoff بهمون میگه اندازه چقدر است. فیلدهای ip-hl و ip-v هرکدام ۴ بیت دارند که در مجموع میشود ۸ بیت یعنی ۱ بایت. لیکن الزامات تراز از نوع زیربنایی به اندازهٔ ساختار تأثیر نگذارد. کامپایلرها اغلب ساختارها را به‌روی مرزهایی تراز می‌کنند که برابر مضرب بزرگ‌ترین نوع اولیه در ساختار هستند. (اینجا می‌شود unsigned int چون ip-hl و ip-v هردو unsigned int هستند) این بدان معناست که حتی با اینکه ۱ بایت داده معنی‌دار داریم، کامپایلر ساختار را به اندازهٔ unsigned int تراز می‌کند یعنی ۴ بایت

پاسخ: ۴ بایت (البته در تعداد خیلی محدودی از کامپایلرها ممکن است تراز کردنی اتفاق نیفتد که در این صورت جواب می‌شود ۱ بایت اما در بیشترشان اینطور نیست)

۷-ب) این نوع تعریف، "bit field" نام دارد که به ما اجازه می‌دهد که بخش‌های کوچکتری از داده‌ها را در یک واحد بزرگتر (مثل یک بایت یا یک word) ذخیره کنیم.

در این مورد، ip_v و ip_hl هردو فقط ۴ بیت از ۸ بیت یا بایت که ۴ بیت اول اگر little edian باشد، به ip_hl و اگر big edian باشد به ip_v تعلق دارد. و ۴ بیت دوم برعکس.

به عبارت واضح‌تر (مخصوص اینجا) اجازهٔ تخصیص تعداد مشخصی از بیت‌ها را برای یک متغیر می‌دهد.

Senobar

ادامه در صفحهٔ بعد

SUBJECT:

Year: Month: Day:

Page : ( )

۷ الف)

شرط‌های قبل از کامپایل شامل

```
# if __BYTE_ORDER == __LITTLE_ENDIAN
...
#endif
#if __BYTE_ORDER == __BIG_ENDIAN
#endif
```

به این دلیل استفاده می‌شوند که نحوهٔ ذخیره‌سازی بایت‌ها در حافظه بسته به نوع معماری سیستم متفاوت است.

این تفاوت‌ها می‌تواند تأثیر مستقیمی بر نحوهٔ دسترسی به داده‌ها در حافظه داشته باشد (نحوهٔ خواندنشان). بنابراین هنگام تعریف ساختارهایی که ممکن است به صورت مستقیم به بایت‌ها دسترسی داشته باشند، باید نحوهٔ ترتیب بایت مشخص شود تا برنامه بتواند به درستی با داده‌ها کار کند.

(Little-endian : بایت کم ارزش‌تر در آدرس کوچک‌تر

Big-endian : بایت پرارزش‌تر در آدرس کوچک‌تر)

در پردازنده‌های Intel که از معماری x86 استفاده می‌کنند، ترتیب بایت به صورت little endian است بنابراین در شرط اول قرار می‌گیرند

Senobar

۶-

با توجه به شماره پروتکل ها در تصویر ارائه شده:

incoming → proto: 6 → نشان دهنده پروتکل TCP است

broadcast → proto: 17 → نشان دهنده پروتکل UDP است

multicast, outgoing → proto: 2 → نشان دهنده پروتکل IGMP (Internet Goup Managemet Protocol) است.

بین ۵ شود TCP و UDP و IGMP

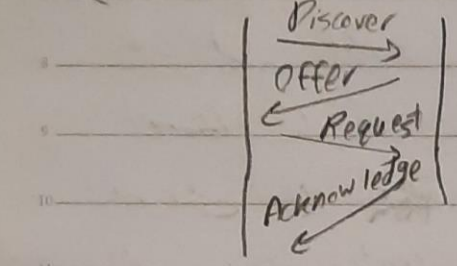
Year: Month: Day: Page: ( )

ادامه ۱ (صفحه سوم سوال ۱)

برای اتصال به شبکه اینترنت

چون لپ تاپ برای دریافت IP addr از DHCP استفاده می کند، پیغام های DHCP نیز که شامل DHCP Discover (درخواست) و پاسخ های DHCP Offer, DHCP Request, DHCP Acknowledge هستند، شناخته می شوند.

(گوشی client) ← server



→ نکته ای که قابل توجه است این است که بین DHCP Request و DHCP Acknowledge ، ARP Request داریم ← دلیلش این است که گوشی قبل از اینکه IP را بپذیرد، یک ARP Request به صورت broadcast ارسال می کند که ببیند آیا host دیگری در شبکه با IP addr داده شده حضور دارد. اگر Reply ای نگیرد یعنی این آدرس، conflict ای با بقیه host ها ندارد و آزاد برای استفاده است.

ادامه ۱ (صفحه دوم سوال ۱):

انتظارمان در خروجی tcpdump:

۱- درخواست های ARP: این درخواست هایی هستند که چیکس دستگاهی که سیستممان می خواهد با آن ارتباط برقرار کند را دارد، به سیستم ما بگوید. (IP addr)

(در حقیقت وقتی ورودی های ARP را حذف می کنیم و سیستم ما نیاز به برقراری ارتباط با هریک از آدرس های IP حذف شده دارد، این درخواست ها را در شبکه پخش می کند)

۲- پاسخ های ARP: اگر دستگاه با آدرس IP درخواستی در همان شبکه موجود باشد، پاسخ ARP را دریافت می کند: IP addr در MAC addr فلان است که فلان، همان است.

ج) در ابتدا بسته های ARP که شامل ARP Request و ARP Reply هستند و برای کشف MAC addr های دستگاه های موجود در شبکه جات اسباب ارسال می شوند را می بینیم.

زمانیکه لپ تاپ ما بخواهد MAC addr router گوشی را بداند درخواست ARP ارسال می کند و ... (پاسخ) MAC addr خودرا ارسال می کند. ادامه در صفحه بعد

۴۰۱۲۱۰۹۳
امیرحسین حامی

SUBJECT:
Year: Month: Day:
Page : ( )

۱-

الف)

```
sudo tcpdump not ip
```

ب) هنگامی که سیستم‌ها نیاز به برقراری ارتباط مجدد با هریک از آدرس‌های IP دارند، باید آدرس‌های MAC متناظر با آن IP که با دستور sudo arp -d <ip_address> فراموش کرده را با ارسال درخواست‌های ARP دوباره یادبگیرد.

خروجی tcpdump:

18:37.00.123456 ARP, Request who-has 192.168.1.10 tell 192.168.1.1, length 33

18:37.00.123791 ARP, Reply 192.168.1.10 is-at 00:11:22:33:44:55, length 20

یک درخواست ARP از 192.168.1.1 نشان می‌دهد که آدرس MAC 192.168.1.10 را درخواست می‌کند

پاسخ ARP از 192.168.1.10 نشان می‌دهد که آدرس MAC خود را ارائه می‌دهد

ادامه در صفحه بعد

SUBJECT
Year: Month: Day:
Page: ( )

ادامه ۲ (صفحه دوم سوال ۲)

در استراکچر sockaddr
_SOCKADDR_COMMON(sa_)
به صورت زیر define شده:

#define _SOCKADDR_COMMON(sa_prefix) \
sa_family_t sa_prefix##family

۳- ۶ بایت (۱۲ رقم هگزا): ac:7b:a1:4f:4c:0f
دوم outgoing frame

۴- 64:70:02:f0:f6:16
۶ بایت (۱۲ رقم هگزا) دوم پاسخ ARP ← ARP incoming

۵- ntoch مخفف "network to host short" است و برای تبدیل یک مقدار ۱۶ بیتی از نحوهٔ ترتیب بایت شبکه به ترتیب بایت host است. منظور از ترتیب بایت، big endian یا little endian بودن داده است. طبیعتاً در شبکه داده‌ها (که به طور معمول، big endian هستند) باید به ترتیب بایت host تبدیل شود وگرنه داده‌ها ممکن است به طور اشتباه موقع رسیدن و در حافظه host پردازش شوند.

ترتیب بایت
به احتمال زیاد

Senobar

مثلاً ممکن است داده‌ها در این صورت، این داده، با هر ترتیب بایتی، همین داده می‌شود

SUBJECT:

Year: Month: Day:

Page : ( )

structure of sockaddr_ll:

```
struct sockaddr_ll{
    unsigned short int sll_family; // Address family
    //          //          // sll_protocol; // Protocol : IPv4
    int sll_ifindex; // interface index
    unsigned short int sll_hatype; // Hardware
    // type : Ethernet → eth1 & eth0 ...
    unsigned char sll_pkttype; // Packet type:
    // Host or Broadcast
    unsigned char sll_halen; // Length of the
    // hard ware address
    unsigned char sll_addr[8]; // Destination
    // MAC address
};
```

structure of sockaddr:

```
struct sockaddr{
    _SOCKADDR_COMMON(sa_);
    /* Common data: address family and length. */
    char sa_data[14];
    /* Address data. */};
```

Senobar