به نام خدا

درس: شبکه مخابرات داده‌ها

استاد: دکتر محمدرضا پاکروان

گزارش پروژه شماره ۲

سیّدمحمّدامین منصوری طهرانی

۹۴۱۰۵۱۷۴

Ethernet Network

۱. مشخص است که اگر هر فریم ارسالی برای رسیدن به دورترین نود در شبکه اترنت مدت t_{prop} زمان بخواهد، اگر فرستنده فریمی بفرستد و در این گیرنده collision رخ دهد نیز پس از گذشتن همان t_{prop} خبر collision به گیرنده می‌رسد. پس اگر مدت زمانی که یک فریم روی کانال است، کمتر از مجموع دو زمان بالا یا $2t_{prop}$ باشد، collision رخ می‌دهد ولی از فرستاده شدن اطلاعات جدید جلوگیری نشده و یک فریم در گیرنده خراب شده‌است. در هر بار به‌دست آوردن دوباره کانال توسط این فرستنده نیز اگر به دورترین گیرنده فریمی بفرستد و collision رخ دهد باز همین اتفاق می‌افتد. اما اگر طول فریم‌ها با سرعت ارسال آن‌ها طوری تنظیم شود که طول زمانی هر فریم از $2t_{prop}$ بیشتر باشد، فرستنده در حین ارسال متوجه collision می‌شود و داده‌ها را در زمان به‌دست آوردن بعدی کانال برای گیرنده از اول می‌فرستد.

۲. از بحث سوال قبل حد زمانی فریم‌ها برای کارکرد صحیح مشخص شد. مشکل به‌وجود آمده افزایش سرعت ارسال است که طول زمانی هر بیت و متناظراً فریم را کاهش داده و به کمتر از حد لازم رسانده‌است. پس باید بیت‌های بیشتر یا معادلاً فریم بزرگتری ارسال کند تا همان زمان مینیمم را در کانال اشغال کند و فرستنده در صورت رخ دادن collision از آن آگاه شود.

$$512bits \times T_{bit} = minimum\ time\ each\ frame\ must\ occupy\ the\ channel$$

$$x\ bits\ \times T_{bit,new} = minimum\ time\ each\ frame\ must\ occupy\ the\ channel$$

می‌دانیم که سرعت ۱۰ برابر شده و معادلاً T_{bit} بر ۱۰ تقسیم شده‌است. لذا طول فریم باید ۱۰ برابر شود و به این ترتیب حداقل طول فریم ارسالی این کاربر ۵۱۲۰ بیت است.

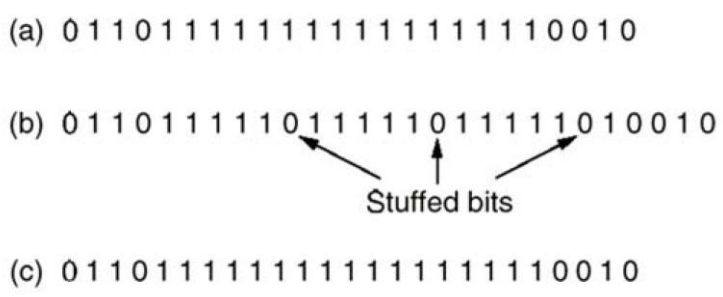
۳. با توجه به این‌که تنها راه حل، افزایش مدت زمان اشغال شده توسط این گیرنده است، یک روش می‌تواند اضافه کردن بافرهایی باشد که فریم با همان طول ۵۱۲ بیت را نگه‌دارند ولی این انتقال از یک بافر به بافر دیگر با تاخیر همراه خواهد بود. لذا تعداد بافرها را به گونه‌ای انتخاب می‌کنیم که مجموع تاخیرها و طول خود فریم، شرط حداقل زمان اشغال کردن کانال را برآورده کنند. در هر بافر نیز ۱۰ فریم نگه می‌داریم. در واقع هر ۱۰ فریم که رسید این‌ها را با هم ارسال می‌کنیم که معادل زیاد کردن زمان propagation خواهد بود. ضمناً فرض می‌کنیم الگوریتم‌های لایه Data Link به خوبی پیاده‌سازی شده‌اند و اگر به خاطر این burst رسیدن فریم‌ها، یکی از آن‌ها خراب بود، گیرنده با روش‌های متداول مانند Selective Repeat از عهده مشکل ترتیب برمی‌آید.

Bit/Byte Stuffing

۱. همان‌طور که می‌دانیم علت فریم‌کردن بیت‌ها گرفتن ack برای بسته‌های داده به جای گرفتن ack برای تک تک بیت‌هاست. در حالتی که مرز بایت‌ها در لایه فیزیکی برای گیرنده مشخص باشد، بین بایت‌ها یک سری بایت نشانه برای مشخص کردن ابتدا و انتهای فریم استفاده می‌شود (Byte Stuffing) و در حالتی که این مرز بین بایت‌ها حفظ نشود، بین بیت‌ها یک الگوی خاص از بیت‌ها به عنوان مشخص کننده ابتدا و انتهای فریم استفاده می‌شود(Bit Stuffing).

در حالت اول (Byte Stuffing) یک کاراکتر خاص به عنوان نشانه یا flag انتخاب می‌شود و گیرنده هر جا آن را مشاهده کرد به عنوان ابتدای فریم و در مشاهده بعدی آن‌را به عنوان انتهای فریم در نظر می‌گیرد. هر وقت نیز نیاز بود در خود داده‌های ارسالی کاراکتر flag ارسال شود، قبل از آن از کاراکتر فرار استفاده می‌شود. اگر در داده خود کاراکتر escape وجود داشت، قبل از آن دوباره یک کاراکتر esc می‌گذاریم. به این ترتیب برای استفاده از کاراکترهای flag یا escape در خود اطلاعات ارسالی قبل از آن‌ها از کاراکتر esc استفاده می‌کنیم و در گیرنده این esc ها برداشته می‌شوند و باقی مانده به عنوان داده تفسیر می‌شود و flag های تنها که قبل آن‌ها esc نیست نیز به عنوان ابتدا و انتهای فریم مشخص می‌شوند.

در حالت دوم (Bit Stuffing)، گیرنده و فرستنده بر روی یک الگو از بیت‌ها توافق دارند و اگر این الگو در خود داده‌های ارسالی موجود بود، با گذاشتن بیت خاصی آن را از شکل الگوی مشخص کننده انتها و ابتدا خارج می‌کنند و در گیرنده اگر این الگو مشاهده شد ابتدا بیت خاص توافق شده در صورت وجود حذف می‌شود و داده‌های اصلی بازیابی می‌شوند. اگر بیت خاص وجود نداشت نیز به منزله ابتدا و سپس انتهای فریم است.(شکل زیر)



در این‌جا الگوی مشخص کننده ابتدا و انتهای فریم ۰۱۱۱۱۱۱۰ بوده و چون در داده‌ها بیش از پنج ۱ پشت سر هم قرار گرفته‌اند، برای جلوگیری از وقوع ششمین ۱ در ادامه آن و احتمالاً یک ۰ که منجر به الگوی تشخیص شود، پس از هر پنج ۱، یک ۰ می‌گذاریم. در گیرنده صفرهای انتهایی پس از هر پنج ۱، حذف می‌شوند و داده اصلی بازیابی می‌شود.

یک پروتکل مشهور که از تکنیک Bit Stuffing استفاده می‌کند، پروتکل لایه دوم SNA است.(مربوط به IBM)(High-level Data Link Control HDLC)

پروتکل PPP (Point to Point protocol) و همچنین UART (Universal Asynchronous receiver-transmitter) از Byte Stuffing استفاده می‌کنند.

۲. اگر فرض کنیم همان الگوی اشاره شده در فوق که در کتاب هم ذکر شده برای bit stuffing استفاده شود، رشته خروجی در لایه فیزیکی به صورت زیر خواهد بود(بیت‌های اضافه شده مشخص شده‌اند.):

01111<u>0</u>01111101111<u>1</u>010

۳. این حالت ماکزیمم زمانی رخ می‌دهد که همه بایت‌های ارسالی از نوع flag یا esc باشند. در این صورت برای هر کدام از این بایت‌ها یک esc استفاده می‌شود. دو بایت نیز در ابتدا و انتها گذاشته می‌شود. بنابراین اگر یک فریم حاوی m بایت باشد، باید $m+2$ بایت overhead به آن اضافه کرد.

Ethernet Protocol

۱. کابل‌های ethernet نسبتاً پیچیده بودند. کابل‌های 10Base5 و 10Base2 که ضخیم و نازک هستند به توپولوژی سیم‌کشی bus نیاز داشتند که در صورت فراتر رفتن از چند node مدیریت آن‌ها پیچیده می‌شود. بدون قطع کردن ارتباط دیگران در کل شبکه امکان اضافه یا حذف node ها نبود. هم‌چنین نقاط جمع شدن زیادی بود که شبکه را در معرض خرابی قرار می‌داد و ضمناً عیب‌یابی آن را نیز دشوار می‌کرد. چیزی که برتری را برای ethernet به ارمغان آورد 10BaseT بود که هم آن‌را ارزان‌تر و هم مدیریت‌پذیر تر از Token Ring می کرد. ویژگی‌های آن عبارت‌اند از:

- توپولوژی ستاره در اطراف hub ها.
- قابلیت نگهداری بیشتر چون مسائل اضافه کردن و حذف کردن گره‌ها تنها به همان گره‌ها محدود شد.
- مقیاس پذیری قوی‌تر در سوییچ کردن

۲. در استاندارد ethernet مقرر شده که بیشینه فاصله دو گره ارتباطی از هم با توجه به ویژگی کابل‌هایی که فضای ethernet را می‌سازند ۵۰۰۰ متر باشد. بنابراین اگر فرض کنیم سرعت انتقال اطلاعات در twisted pair $\frac{۲}{۳}$ سرعت نور است.(سرعت انتقال ۲×۱۰^{۸}):

$$2t_{prop} = 5000 \times \frac{2}{v} = numOfBits \times T_{bit}$$

$$v = 2 \times 10^8, T_{bit} = (10M\text{b/sec})^{-1} \rightarrow numOfBits = 10000 \times \frac{10^7}{2 \times 10^8} = 500$$

بنابراین یک مضرب دو مناسب ۵۱۲ بیت یا ۶۴ بایت است.

۳. در انتهای دیگر محدوده ۱۵۰۰ بایت، دو عامل وجود دارد که منجر به معرفی این حد می شود. اول، اگر بسته‌ها بیش از حد طول بکشند، با استفاده از کابل اترنت، تاخیرهای بیشتری را به ترافیک دیگران اضافه می کنند. عامل دیگر یک دستگاه ایمنی ساخته شده در فرستنده‌های کابلی اولیه بود. این دستگاه ایمنی یک سیستم ضد پرحرفی بود. اگر دستگاه متصل به یک فرستنده و گیرنده یک خطا ایجاد کرده و به طور مداوم ارسال می‌کرد، به طور موثر هر نوع ترافیک دیگر را از استفاده از بخش کابل اترنت مسدود می کند. برای جلوگیری از این اتفاق، فرستنده های اولیه برای خاموش شدن به صورت خودکار در صورتی‌که انتقال بیش از ۱/۲۵ میلی ثانیه باشد طراحی شده‌بودند. این معادل محتوای داده بیش از ۱۵۰۰ بایت است. با این حال، در حالی‌که فرستنده و گیرنده از یک تایمر آنالوگ ساده برای خاموش کردن انتقال در صورت تشخیص پر حرف بودن استفاده می‌کرد، پس از آن حد ۱۵۰۰، حد مجاز به عنوان یک تقریب صحیح برای حداکثر اندازه داده انتخاب شد.

۴. اگر CRC نامعتبر باشد، فریم به دست آمده دور ریخته می‌شود و درخواست ارسال مجدد خواهیم داشت. فرقی ندارد که این اشکال در payload باشد یا در خود CRC bits.

۵. همان‌طور که در درس دیده‌ایم، هر فریمی که از یک node ارسال می‌شود باید مانند نامه حاوی آدرس فرستنده آن و گیرنده آن باشد. MAC Address همان آدرس فرستنده/گیرنده است و برای شناختن آن‌ها استفاده می‌شود. هر دستگاه(اعم از تلفن همراه، لپتاپ و به طور کلی هر وسیله‌ای که به شبکه اینترنت وصل می‌شود) دارای یک آدرس است و به آن MAC Address می‌گویند. OUI یا organizationally unique identifier یک عدد ۲۴ بیتی است که بخشی از MAC Address بوده (۳ octet اول) و تولیدکننده/فروشنده/سازمان خاصی را مشخص می‌کند. این آدرس‌ها از IEEE خریداری می‌شوند.

Cyclic Redundancy Check

۱. در این روش تعداد مشخصی بیت به آخر فریم برای تشخیص خطا اضافه می‌شود. از مزایای کدهای چرخشی مناسب بودن آن‌ها برای تشخیص خطای burst است(خطای پشت سرهم) و این خطاها در بسیاری از کانال‌های مخابراتی رخ می‌دهند. هم‌چنین با اضافه کردن یک کد n CRC بیتی قادر خواهیم بود تمام تک خطاهای burst با طول کمتر n و کسر $1 - 2^{-n}$ از تمام خطاهای burst با طول بیشتر را تشخیص دهیم که قدرت زیادی را در اختیارمان قرار می‌دهد.

در این روش بیت‌های داده که قرار است ارسال شوند به منزله ضرایب یک چندجمله‌ای تعبیر می‌شوند. مثلاً یک رشته ۱۰ بیتی(۱۱۰۱۰۱۱۰۱۱) شامل ۱۰ ضریب است پس چندجمله‌ای از درجه ۹ را توصیف می‌کند.

$$M(x) = 1 \cdot x^9 + 1 \cdot x^8 + 0 \cdot x^7 + 1 \cdot x^6 + 0 \cdot x^5 + 1 \cdot x^4 + 1 \cdot x^3 + 0 \cdot x^2 + 1 \cdot x^1 + 1 \cdot x^0$$
$$M(x) = x^9 + x^8 + x^6 + x^4 + x^3 + x^1 + 1$$

برای محاسبه CRC به یک چندجمله‌ای دیگر نیز نیاز داریم و آن را چندجمله‌ای مولد می‌خوانیم و درجه آن باید بیشتر از صفر و کمتر از چندجمله‌ای $M(x)$ باشد. با توجه به گزینه‌های مختلف ممکن برای این چندجمله‌ای از استاندارد برای آن استفاده می‌شود. برای مثال در CRC-32 چندجمله‌ای مولد به شکل زیر است.

$$G(x) = x^{32} + x^{26} + x^{23} + x^{22} + x^{16} + x^{12} + x^{11} + x^{10} + x^8 + x^7 + x^5 + x^4 + x^2 + x + 1$$

در حالت کلی برای محاسبه n CRC بیتی به این صورت عمل می‌شود که رشته داده به چند جمله‌ای $M(x)$ تعبیر شده و در x^n ضرب می‌شود(که n درجه چندجمله‌ای مولد نیز هست.) و بر چندجمله‌ای مولد تقسیم می‌شود. باقی‌مانده تقسیم به همراه داده ارسال می‌شود و در گیرنده همین رشته ارسالی دوباره بر چندجمله‌ای مولد تقسیم شده و اگر خطا رخ نداده باشد باقی مانده صفر خواهد بود و اگر باقی‌مانده صفر نباشد خطا در جایی از داده دریافتی نتیجه گرفته می‌شود. تعریف جمع و تقسیم و ضرب چندجمله‌ای‌ها در مقاله گفته شده و از آوردن دوباره آن‌ها خودداری می‌کنیم.

۲. نتیجه جستجو در ویکی‌پدیا در زیر آورده شده‌است. CRC ۶ بیتی در شبکه‌های موبایل کاربرد دارد. همین‌طور ۸ و ۱۰ و ۱۲ و ۱۴. ۳۰ بیتی در Code Division Multiple Access استفاده می‌شود. ۳۲ بیتی به نظر می‌آید از همه کاربردهای بیشتری دارد. از جمله کاربردهای اصلی آن Ethernet است.

CRC-8-WCDMA	mobile networks[25][35]
CRC-10	ATM; ITU-T I.610
CRC-10-CDMA2000	mobile networks[25]
CRC-10-GSM	mobile networks[22]
CRC-11	FlexRay[36]
CRC-12	telecom systems[37][38]
CRC-12-CDMA2000	mobile networks[25]
CRC-12-GSM	mobile networks[22]
CRC-13-BBC	Time signal, Radio teleswitch[39][40]
CRC-14-DARC	Data Radio Channel[26]
CRC-14-GSM	mobile networks[22]
CRC-15-CAN	
CRC-15-MPT1327	[43]
CRC-16-Chakravarty	Optimal for payloads ≤64 bits[28]
CRC-16-ARINC	ACARS applications[44]
CRC-16-CCITT	X.25, V.41, HDLC *FCS*, XMODEM, Bluetooth, PACTOR, SD, DigRF, many others; known as *CRC-CCITT*
CRC-16-CDMA2000	mobile networks[25]
CRC-16-DECT	cordless telephones[45]
CRC-16-T10-DIF	SCSI DIF
CRC-16-DNP	DNP, IEC 870, M-Bus
CRC-16-IBM	Bisync, Modbus, USB, ANSI X3.28, SIA DC-07, many others; also known as *CRC-16* and *CRC-16-ANSI*
CRC-16-OpenSafety-A	safety fieldbus[32]
CRC-16-OpenSafety-B	safety fieldbus[32]

Name	Uses
CRC-1	most hardware; also known as *parity bit*
CRC-3-GSM	mobile networks[22]
CRC-4-ITU	ITU-T G.704, p. 12
CRC-5-EPC	Gen 2 RFID[24]
CRC-5-ITU	ITU-T G.704, p. 9
CRC-5-USB	USB token packets
CRC-6-CDMA2000-A	mobile networks[25]
CRC-6-CDMA2000-B	mobile networks[25]
CRC-6-DARC	Data Radio Channel[26]
CRC-6-GSM	mobile networks[22]
CRC-6-ITU	ITU-T G.704, p. 3
CRC-7	telecom systems, ITU-T G.707, ITU-T G.832, MMC, SD
CRC-7-MVB	Train Communication Network, IEC 60870-5[28]
CRC-8	DVB-S2[29]
CRC-8-AUTOSAR	automotive integration,[31] OpenSafety[32]
CRC-8-Bluetooth	wireless connectivity[33]
CRC-8-CCITT	ITU-T I.432.1 (02/99); ATM HEC, ISDN HEC and cell delineation
CRC-8-Dallas/Maxim	1-Wire bus[34]
CRC-8-DARC	Data Radio Channel[26]
CRC-8-GSM-B	mobile networks[22]
CRC-8-SAE J1850	AES3

CRC-16-Profibus	fieldbus networks[47]
Fletcher-16	Used in Adler-32 A & B Checksums
CRC-17-CAN	CAN FD[48]
CRC-21-CAN	CAN FD[48]
CRC-24	FlexRay[36]
CRC-24-Radix-64	OpenPGP, RTCM104v3
CRC-30	CDMA
CRC-32	ISO 3309 (HDLC), ANSI X3.66 (ADCCP), FIPS PUB 71, FED-STD-1003, ITU-T V.42, ISO/IEC/IEEE 802-3 (Ethernet), SATA, MPEG-2, PKZIP, Gzip, Bzip2, POSIX cksum,[49] PNG,[50] ZMODEM, many others
CRC-32C (Castagnoli)	iSCSI, SCTP, G.hn payload, SSE4.2, Btrfs, ext4, Ceph
CRC-32K (Koopman {1,3,28})	Excellent at Ethernet frame length, poor performance with long files
CRC-32K$_2$ (Koopman {1,1,30})	Excellent at Ethernet frame length, poor performance with long files
CRC-32Q	aviation; AIXM[51]
Adler-32	
CRC-40-GSM	GSM control channel[52][53][54]
CRC-64-ECMA	ECMA-182 p. 51, XZ Utils
CRC-64-ISO	ISO 3309 (HDLC), Swiss-Prot/TrEMBL; considered weak for hashing[55]

۳. نکته مهم: چیزی که به عنوان CRC-32 در استاندارد IEEE 802.3 شناخته می‌شود به صورت زیر است:

چندجمله‌ای $M(x)$ شامل آدرس مقصد، مبدا، طول داده، خود داده فریم(۳۲ بیت اول همه این‌ها متمم شده‌است.) است. باقی‌مانده تقسیم نیز متمم شده و به انتهای فریم اضافه می‌شود.(CRC-32)

ضمناً در محاسبه CRC برای فریم طبیعتاً ۳۲ بیت آخر که خود CRC است را در نظر نمی‌گیریم.

نتیجه خروجی کد صفر شد و اشتباه است(باید E6C53DB2 باشد.). کد فلیپ فلاپ را شبیه سازی کرده ولی متوجه نشدیم مشکل چیست.

Investigating an Ethernet Interface

۱. از Wireless LAN استفاده می‌کنیم. نتیجه در زیر آورده شده‌است. چون این دستور به ethernet مربوط است و ما از WLAN استفاده کرده‌ایم، خروجی سرعت ethernet برای سیستم قابل اندازه‌گیری نبوده و مقدار unknown را برمی‌گرداند.(خط ۱۴ تصویر زیر)

```
ehsan27770@ehsan27770-Studio-XPS-1340:~$ sudo ethtool eth0
Settings for eth0:
        Supported ports: [ MII ]
        Supported link modes:   10baseT/Half 10baseT/Full
                                100baseT/Half 100baseT/Full
                                1000baseT/Full
        Supported pause frame use: No
        Supports auto-negotiation: Yes
        Advertised link modes:  10baseT/Half 10baseT/Full
                                100baseT/Half 100baseT/Full
                                1000baseT/Full
        Advertised pause frame use: No
        Advertised auto-negotiation: Yes
        Speed: Unknown!
        Duplex: Unknown! (255)
        Port: MII
        PHYAD: 0
        Transceiver: external
        Auto-negotiation: on
        Supports Wake-on: g
        Wake-on: d
        Link detected: no
ehsan27770@ehsan27770-Studio-XPS-1340:~$
```

۲.

- MAC Address یک عدد ۶ بایتی یا ۴۸ بیتی است و در تصویر زیر در خط دوم قابل مشاهده است. (توجه می‌کنیم که اطلاعات eth مربوط به دستگاه ما است و اطلاعات wlan0 مربوط به ماژول فرستنده است برای همین MAC Address نوشته شده در آن‌جا با MAC Address قسمت ethernet تفاوت دارد.)

$$00:22:19:e9:cc:22$$

```
ehsan27770@ehsan27770-Studio-XPS-1340:~$ ifconfig
eth0      Link encap:Ethernet  HWaddr 00:22:19:e9:cc:22
          UP BROADCAST MULTICAST  MTU:1500  Metric:1
          RX packets:0 errors:0 dropped:0 overruns:0 frame:0
          TX packets:0 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:0 (0.0 B)  TX bytes:0 (0.0 B)

lo        Link encap:Local Loopback
          inet addr:127.0.0.1  Mask:255.0.0.0
          inet6 addr: ::1/128 Scope:Host
          UP LOOPBACK RUNNING  MTU:65536  Metric:1
          RX packets:919 errors:0 dropped:0 overruns:0 frame:0
          TX packets:919 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1
          RX bytes:69580 (69.5 KB)  TX bytes:69580 (69.5 KB)

wlan0     Link encap:Ethernet  HWaddr 00:25:56:6b:98:e0
          inet addr:192.168.43.219  Bcast:192.168.43.255  Mask:255.255.255.0
          inet6 addr: fe80::225:56ff:fe6b:98e0/64 Scope:Link
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:11301 errors:0 dropped:0 overruns:0 frame:0
          TX packets:6881 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:15418620 (15.4 MB)  TX bytes:659086 (659.0 KB)

ehsan27770@ehsan27770-Studio-XPS-1340:~$
```

- با جستجو در اینترنت از یک OUI Lookup آنلاین استفاده می‌کنیم و با دادن ۶ بایت اول vendor یا فروشنده لپتاپمان را می‌یابیم(که مطابق انتظار است!):

Results for MAC address 00:22:19

Found 1 results.

MAC Address/OUI	Vendor {Company}
00:22:19	Dell Inc.

- در تصویر فوق در ماژول فرستنده و همچنین ethernet، MTU و مقدار آن قابل مشاهده‌اند.(۱۵۰۰)
- در قسمت WLAN عدد RX برابر ۱۱۳۰۱ packet یا ۱۵/۴ مگابایت است.
- در قسمت WLAN عدد TX برابر ۶۸۸۱ packet یا ۶۵۹ کیلوبایت است.
- خیر. هیچ collision ای موجود نیست و در خط یکی مانده به آخر تصویر بالا عدد صفر مشاهده می‌شود.
- خیر. هیچ packet ای نیز دچار مشکل نشده و دور ریخته نشده است. در سومین و چهارمین خط از آخر این نتیجه قابل مشاهده است.

۳. علت آن یکی می‌تواند این باشد که در محیطی که از WLAN استفاده می‌کنیم فقط یک کاربر هستم و با کاربر دیگری نمی‌توان collision داشت. علت دور ریخته نشدن هیچ packet نیز می‌تواند این باشد که مدت زمان اتصال نسبتاً کم بوده و چون احتمال خراب شدن و در نتیجه دور ریخته شدن کم است، در این مدت هیچ دور ریخته شدنی مشاهده نشده‌است.

Exploring Ethernet Frames

۱. Packet انتخاب شده در فایل ارسال شده ضمیمه شده‌است.

۲. برای packet های مختلف نتیجه فرق دارد.

برای یک packet از نوع TCP:

```
Ethernet II, Src: IntelCor_c0:21:cd (48:45:20:c0:21:cd), Dst: Cisco_ef:f4:40 (00:27:0d:ef:f4:40)
```

MAC Address/OUI	Vendor {Company}
00:27:0D	Cisco Systems, Inc

MAC Address/OUI	Vendor {Company}
48:45:20	Intel Corporate

برای یک packet از نوع UDP:

```
Ethernet II, Src: AsustekC_7e:82:2a (38:d5:47:7e:82:2a), Dst: Broadcast (ff:ff:ff:ff:ff:ff)
```

MAC Address/OUI	Vendor {Company}
38:D5:47	ASUSTek COMPUTER INC.

این packet از نوع broadcast است و برای همین آدرس مقصد ff:ff:ff:ff:ff:ff است. آدرس مبدا هم مربوط به لپتاپ است که ASUS است.(در قسمت قبل به علت مشکل پیدا کردن Linux در لپتاپم از لپتاپ دیگری استفاده کردم.)

برای یک packet از نوع ARP:

```
Ethernet II, Src: AsustekC_c0:d6:d0 (48:5b:39:c0:d6:d0), Dst: Broadcast (ff:ff:ff:ff:ff:ff)
```

MAC Address/OUI	Vendor {Company}
48:5B:39	ASUSTek COMPUTER INC.

۳. TCP:

```
Type: IPv4 (0x0800)
```

UDP:

```
Type: IPv4 (0x0800)
```

ARP:

```
Type: ARP (0x0806)
```

۴. Address Resolution Protocol: به طور کلی دستگاه‌های مختلف به هنگام ارتباط در شبکه به وسیله آدرس IP با هم در ارتباط‌اند و آن‌را برای هم می‌فرستند. اما برای ارسال در لایه فیزیکی به MAC Address نیاز است. وقتی یک بسته قرار است برای کسی ارسال شود، در ابتدا فرستنده از برنامه ARP آدرس فیزیکی گیرنده‌ای که IP آن را دارد می‌پرسد(MAC Address). برنامه ARP در ARP cache جستجو می‌کند و اگر این MAC Address در حافظه موجود بود با قرار دادن آن در جای مناسب فریم، آن‌را ارسال می‌کند. اگر این آدرس یافت نشد فرستنده یک پیام broadcast با آدرس مقصد ff:ff:ff:ff:ff:ff در محدوده LAN ارسال می‌کند و MAC Address مربوط به IP مورد نظرش را از صاحب آن درخواست می‌کند. دستگاهی که صاحب IP است این آدرس را برای فرستنده می‌فرستد و این آدرس برای مراجعات بعدی در lookup table مربوط به ARP نیز ذخیره می‌شود.

از قسمت قبل برای packet از نوع ARP داشتیم:

```
Ethernet II, Src: AsustekC_c0:d6:d0 (48:5b:39:c0:d6:d0), Dst: Broadcast (ff:ff:ff:ff:ff:ff)
```

که مطابق آن‌چه در بالا به آن اشاره شد مقصد آن همه دستگاه‌های در محدوده شبکه هستند و broadcast با آدرس اختصاص داده شده به این کار است.

۵. نرم افزار Wireshark، packet ها را از برنامه دیگری که به خود دستگاه مربوط است می‌گیرد و نکته این‌جاست که خود آن برنامه ابتدا CRC ها را چک کرده و فریم‌های خراب را دور می‌ریزد و بنابراین فریم‌های باقی مانده که Wireshark، capture می‌کند دیگر CRC نخواهند داشت.