پروژه مدارهای منطقی و سیستم‌های دیجیتال

نگارنده:

امیرمهدی سلیمانی‌فر

شماره دانشجویی:

۹۸۱۰۱۷۴۷

استاد درس:

دکتر شاه منصوری

تاریخ تحویل: 21 بهمن ۱۳۹۹

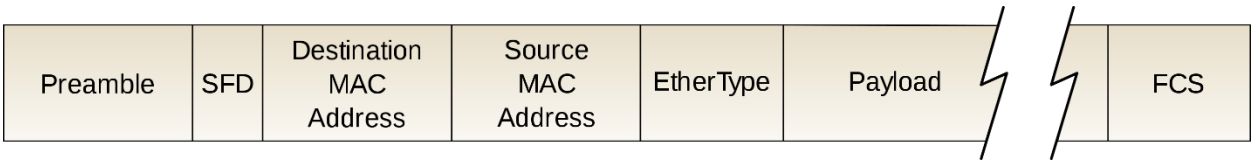
۱۳۹۹-۰۰

توضیحات پروژه

پیاده‌سازی لایه data-link پروتکل Ethernet با وریلاگ

Ethernet روشی ارتباطی برای اتصال کامپیوترهای موجود در یک شبکه‌ی محلی (Local area network) می‌باشد، که داده‌ها در قابل یک فریم به صورت سریال تبادل پیدا می‌کنند. اطلاعات باید در قالب مشخصی منتقل شوند که بتوان در مقصد متوجه زمان ورود اطلاعات جدید شد، و این که آیا اطلاعات درست هستند یا در مسیر خطایی به وجود آمده و مشخصات اطلاعات ورودی مشخص شوند. به این قالب کلی فریم (Frame) می‌گویند. هدف در این پروژه پیاده‌سازی فریم Ethernet به زبان وریلاگ می‌باشد.

در شکل زیر فریم Ethernet در حالت عمومی مشخص شده است:



یک Packet در Ethernet دارای بخش‌های متفاوتی است که توضیحات و کاربرد هر بخش در سوالات تشریحی زیر آورده شده است:

سوالات تشریحی

۱- بخش Preamble در ساختار Ethernet چند بایت است و چه کاربردی دارد؟

بخش Preamble (مقدمه) شامل ۷ بایت (۵۶ بیت) از یک و صفرهای متناوب است، که با توجه به اینکه Ethernet یک سیگنال دیجیتال خود تنظیم (Self-clocked) است و کارکرد آن ارتباطی با کلاک‌های موجود در فرستنده و دریافت کننده ندارد، به دریافت کننده (Receiver) این اجازه را می‌دهد که خود را با کلاک موجود همگام کند تا در مراحل دریافت سیگنال اطلاعات مشکلی بوجود نیاید. به معنای دیگر کلاک در سیگنال‌های اطلاعاتی Ethernet از خود سیگنال استخراج می‌شود.

یک نمونه از از Preamble در با فرمت Hexadecimal در زیر آورده شده است:

0x55 0x55 0x55 0x55 0x55 0x55 0x55

۲- بخش SFD چند بایت است و چه کاربردی دارد؟ آیا با ارسال هر فریم این مقدار تغییری می‌کند؟

SFD که مخفف Start frame delimiter یا محدود کننده شروع فریم است یک بایت (۸ بیت) طول دارد که پایان بخش Preamble (که برای Sync کلاک استفاده می‌شد.) را مشخص می‌کند.

در واقع بخش‌های Preamble و SFD با هم بخش اول Ethernet packet بوده و پس از آن وارد بخش Ethernet frame می‌شویم که حاوی اطلاعات موردنظر ما می‌باشد. هدف از طراحی SFD نشان دادن شکسته شدن الگوی صفر و یک‌های متناوب است که به وسیله آن دستگاه متوجه شروع شدن Ethernet frame خواهد شد. مقدار SFD براساس استاندارد IEEE 802.3 ثابت بوده و همواره بصورت 10101011 تنظیم می‌شود. در بعضی موارد بخش SFD را بعنوان بخشی از Preamble قلمداد می‌کنند و در این نمایش‌ها Preamble دارای ۸ بایت خواهد بود. هدف از سیگنال SFD علاوه بر اینکه نمایش‌دهنده شروع بخش اطلاعات است این می‌باشد که به دستگاه هشدار دهد این آخرین زمان برای همگام‌سازی کلاک است.

۳- کاربرد قسمت‌های Destination MAC Address و Source MAC Address را بیان کرده و تعداد بایت‌های هر کدام را مشخص نمایید.

شروع (Header) بخش Ethernet frame با دو آدرس MAC که به معنای کنترل دسترسی مدیوم/محتوا (Media/medium access control) می‌باشدکه هرکدام دارای ۶ بایت (۴۸ بیت) اطلاعات می‌باشند. MAC آدرس‌ها یک عدد ۱۲ رقمی Hexadecimal هستند که اکثرا با علامت دو نقطه در میان هر دو رقم نمایش داده می‌شوند. ۶ رقم اول تولید کننده MAC address را مشخص می‌کنند که به آنها OUI یا بصورت کامل Organizational Unique Identifier گفته می‌شود. IEEE مسئول ثبت ارقام برای تولید کننده‌های MAC address است. OUI چند تولید کننده بنام بشرح زیر است:

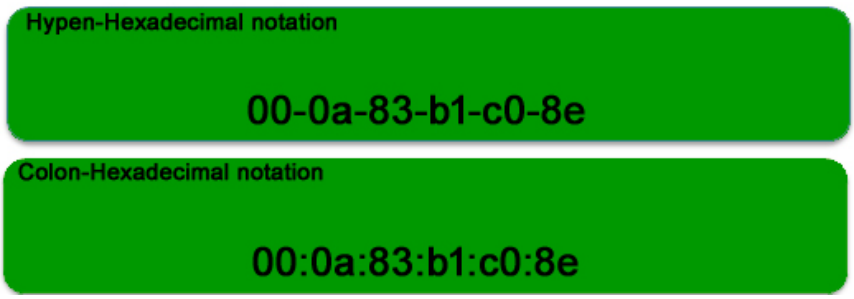
CC:46:D6 - Cisco

3C:5A:B4 - Google, Inc.

3C:D9:2B - Hewlett Packard

00:9A:CD - HUAWEI TECHNOLOGIES CO.,LTD

۶ رقم سمت راست نیز نشان‌دهنده کنترل کننده رابط شبکه (Network Interface Controller) هستند که توسط سازنده به دستگاه انتساب داده می‌شوند. بطور کلی یک MAC address می‌تواند به هر یک از صورت‌های زیر نمایش داده شود:



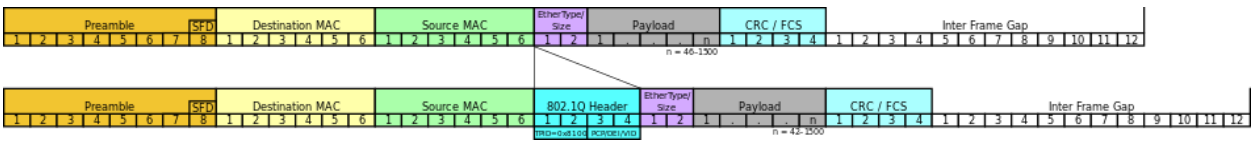
با توجه به اینکه MAC address های مرتبط با هریک از دستگاه‌ها با یکدیگر متفاوت است به این ترتیب می‌توان به وسیله آدرس های موجود فرستنده و دریافت کننده اطلاعات را در شبکه محلی مشخص کرد. با استفاده از دستورهای متفاوت مانند ipconfig /all در ویندوز می‌توان به MAC address دستگاه دسترسی پیدا کرد که برای سیستم من بصورت E4-E7-49-36-32-24 است. (۶ رقم ابتدایی ثبت شده شرکت HP می‌باشند.)

۴- EtherType برای چه مواردی استفاده می‌شود و چند بایت است؟

طول که فیلد EtherType ۲ بایت (۱۶ بیت) است که می‌تواند برای دو کاربرد متفاوت استفاده شود. EtherType برای تعیین پروتکلی که Payload بوسیله آن کپسوله شده است مورد استفاده قرار می‌گیرد. این تعیین پروتکل توسط Receiver برای تعیین آنکه داده دریافتی باید چگونه توسط پردازش شود مورد استفاده قرار می‌گیرد. EtherType اولین بار توسط استاندارد Ethernet II معرفی شده و دو بازه از مقادیر را می‌پذیرد. مقدارهای کوچکتر از ۱۵۰۰ می‌تواند باری تعیین تعداد بایت‌های Payload مورد استفاده قرار گیرند. مقدارهای بزرگتر از ۱۵۳۶ نیز نوع خاصی از EtherType را مشخص می‌کنند برای مثال مقدار 0x0800 نشان‌دهنده‌ی IPv4 یا همان سیستم آدرس دهی اینترنتی نسل ۴ است. به همین ترتیب IPv6 نیز با 0x86DD نمایش داده می‌شود. به این ترتیب ما بوسیله EtherType اطلاعاتی در رابطه با Payload بدست می‌آوریم.

۵- با توجه به متغیر بودن تعداد بایت‌های داده‌ی موجود در Payload چگونه می‌توان متوجه شد که اندازه‌ی هر فریم داده چقدر می‌باشد؟

مقدار حداقلی Payload در صورت وجود تگ 802.1Q حداقل ۴۲ بایت و در صورت عدم وجود آن ۴۶ بایت است و همچنین حداکثر مقدار Payload نیز ۱۵۰۰ بایت است. با این وجود مقدار دقیق تعداد بایت‌های اطلاعات همانگونه که در قسمت قبلی توضیح داده شد از مقدار EtherType بدست می‌آید بصورتی که اگر مقدار آن کمتر از ۱۵۰۰ باشد مقدار نشان‌دهنده تعداد بایت‌های Payload است و در صورتی که این مقدار بیشتر باشد با توجه به اینکه ورودی‌های خاص نشان‌دهنده پروتکل‌های خاص انتقال هستند، بوسیله استانداردهای مرتبط با آنها می‌توان مقدار مربوط به Payload را تشخیص داد.



۶- بخش آخر هر فریم که به FCS اختصاص دارد چه کاربردی دارد و چند بایت را به خود اختصاص می‌دهد؟

دنباله بررسی فریم FCS که در مخفف Frame Check Sequence است یک کد تشخیص خطا است که از ۴ بایت CRC (Cyclic redundancy check) تشکیل شده است.

CRC در واقع می‌تواند داده خراب شده را در Packet دریافت شده توسط Receiver تشخیص دهد. نحوه عملکرد به این ترتیب است که FCS شامل مقداری است که در مبدا براساس داده موجود در فریم که آماده ارسال است محاسبه شده و در انتهای فریم همراه با آن فرستاده شده است. زمانی که گره مقصد فریم را دریافت می‌کند دوباره اقدام به محاسبه FCS براساس رابطه مورد توافق دو انتهای انتقال اطلاعات پرداخته و در صورتی که مقدار محاسبه شده FCS در مقصد با مقداری که در مبدا محاسبه شده بود متفاوت باشند فریم نادیده گرفته می‌شود. به این ترتیب FCS یک بررسی کننده انتقال یا عدم انتقال صحیح اطلاعات است.

۷- استاندارد تشخیص خطای CRC32 را توضیح دهید و با یک مثال و محاسبه نشان دهید که چگونه می‌توان از آن برای تشخیص خطا استفاده نمود؟

در انتقال و فشرده‌سازی اطلاعات فشرده‌سازی فایل و بازیابی خطا دو هدف در مقابل هم هستند، به گونه‌ای که در فشرده‌سازی تمامی بیت‌ها در هم تنیده شده و به نوعی اندازه فایل کاهش می‌یابد اما به همین دلیل اخلال در انتقال و یا ذخیره یکی از بیت‌ها برخلاف حالت عادی که بر یک نتیجه تاثیرگذار بود می‌تواند باعث تغییر زیاد نتیجه شود که برای رفع آن می‌توان از Parity files (که دقیقا بیت‌هایی که دچار خطا شده‌اند را نشان می‌دهد و بوسیله آن می‌توان فایل با خطا را بطور کامل بازیابی کرد.) و یا Check values (که تنها انتقال یا عدم انتقال سالم فایل را نشان می‌دهند.) استفاده کرد. در انتقال اطلاعات با توجه به اینکه می‌توان در صورت تشخیص خطا درخواست ارسال دوباره از فرستنده را داشت روش Check values بهینه‌تر است.

یک روش استفاده از مقدار جمع یا Check sum برای رشته اطلاعات است که البته بدلیل عدم تشخیص بسیاری از خطاها مناسب نیست. یه متد فویتر CRC است که در بالا مختصرا توضیح داده شد. در اینجا بجای جمع از باقیمانده استفاده می‌کنیم و باقیمانده تقسیم چندجمله‌ای به پیمانه ۲ ورودی را بدست می‌آوریم. چند جمله‌ای در حالت عمومی فرم زیر را دارد:

$$c_n x^n + c_{n-1} x^{n-1} + \cdots + c_2 x^2 + c_1 x + c_0$$

همچنین نمونه‌ای از تقسیم چندجمله‌ای ها بر یکدیگر در پایین نمایش داده شده است:

$$\begin{array}{r|l} & 3x^2 + 2x + 5 \\ \hline 4x^3 + 5x^2 + 9x + 12 & 12x^5 + 23x^4 + 57x^3 + 79x^2 + 69x + 60 \\ & \underline{- 12x^5 + 15x^4 + 27x^3 + 36x^2} \\ & 8x^4 + 30x^3 + 43x^2 + 69x \\ & \underline{- 8x^4 + 10x^3 + 18x^2 + 24x} \\ & 20x^3 + 25x^2 + 45x + 60 \\ & \underline{- 20x^3 + 25x^2 + 45x + 60} \end{array}$$

همچنین منظور از چندجمله‌ای به پیمانه ۲ آن است که تمامی ضرایب باید بر دو تقسیم شده و باقیمانده آن‌ها در این تقسیم بجای ضریب قرار بگیرد. به این ترتیب مقدار هر یک از ضرایب از میان صفر و یک است. در اینحالت اعمال جمع و تفریق نتایج یکسانی خواهند داشت و می‌توان در تقسیم به جای تفریق از جمع عبارت‌های جبری استفاده کرد:

a	b	a + b	(a + b) % 2
0	0	0	0
0	1	1	1
1	0	1	1
1	1	2	0

Table 1: addition operation modulo 2

a	b	a - b	(a - b) % 2
0	0	0	0
0	1	-1	1
1	0	1	1
1	1	0	0

Table 2: subtraction operation modulo 2

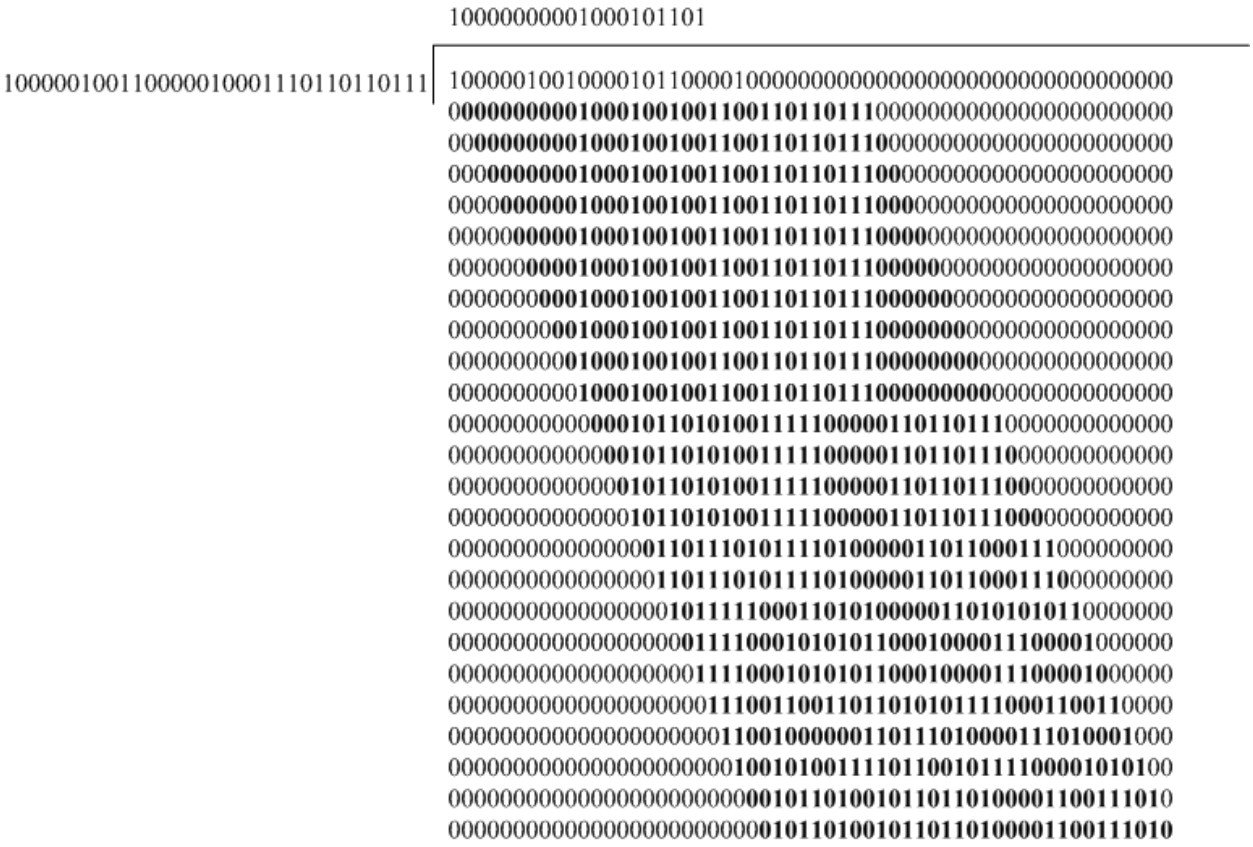
CRC در واقع مخصوصا برای تشخیص خطاهایی طراحی شده است که Checksum قادر به پیداکردن آن‌ها نیست. طراحی الگوریتم CRC به اینگونه بوده است که با رشته‌های طولانی از صفر و یک تطابق داشته باشد و احتمال تشخیص خطا در این موارد به اندازه قابل قبول بالا است.

معمول ترین خارج قسمت مورد استفاده در CRC که به CRC32 معروف است بصورت زیر است:

$$x^{32} + x^{26} + x^{23} + x^{16} + x^{12} + x^{11} + x^{10} + x^8 + x^7 + x^5 + x^4 + x^2 + x + 1$$

در صورتی که عبارتی که می‌خواهیم CRC32 مربوط به آن را محاسبه کنیم باید از کم ارزش‌ترین یک موجود در رشته اطلاعات به اندازه‌ای صفر به سمت راست رشته اضافه کنیم که مقدار پس از آن نیز بزرگتر از چند جمله‌ای خارج قسمت از لحاظ توانی باشد، زیرا در غیر اینصورت پیش از آنکه به تعداد بیت مورد نیاز برای CRC برسیم، عمل تقسیم خاتمه خواهد یافت. با توجه به اینکه اعمال بالا (همان تفریق چندجمله‌ای‌ها) معادل گیت XOR عمل می‌کند لذا زمانی که ما تمامی یک‌های ورودی را از میان ببریم در واقع تقسیم انجام شده است و ما به CRC مورد نظر خود دست پیدا کرده‌ایم. یک تقسیم برای محاسبه CRC32 در صفحه بعدی آورده شده است:

در این تقسیم عبارت خارج قسمت همان چند جمله‌ای CRC32 است که در بالا نمایش داده شده است، همچنین ورودی‌ها به صورت Reverse یا از انتها به ابتدا تحویل داده شده‌اند یعنی در واقع آنچه ما می‌خواهیم CRC آن را حساب کنیم، مقادیر ۸ بیتی ۶۵، ۶۶ و ۶۷ است که در سمت چپ آنها تعداد ۲۴ صفر برای صحت آورده شده است. در اینجا همانگونه که گفته شد تمامی عبارات Reverse هستند اما در پیاده‌سازی وریلاگ آنها را بطور معمول و با در نظرگرفتن ارزش مکانی پیاده‌سازی کرده‌ایم.



می‌دانیم که براساس استانداردها ورودی باید حداقل ۴۲ بایت یا در موارد ۴۶ بایت باشد، اما به دلیل آنکه بررسی صحت کد وریلاگ در این حالت بسیار سخت خواهد شد، از این محدودیت صرف نظر کرده و کد خود را با تست کیس بالا بررسی خواهیم کرد. (دقت شود که در صورت نیاز به اعمال محدودیت می‌توان با استفاده از یک دستور شرطی به راحتی آن را بررسی کرد.)

کد وریلاگ

با توجه به پرسشی که شد می‌دانیم که طول Payload از ۱۵۰۰ بیشتر نخواهد شد و در واقع مقدار EtherType همواره طول Payload را به ما خواهد داد به همین دلیل حداکثر به ۱۵۰۰ بایت یعنی ۱۲۰۰۰ بیت نیاز خواهیم داشت. برای پیاده‌سازی می‌خواهیم تمام یک فریم Ethernet را دریافت کنیم بنابراین از ابتدا بترتیب شروع به دریافت قسمت‌های مختلف می‌کنیم. با توجه به اینکه کلاک خارجی است قسمت‌های Preamble و SFD تغییری را در رشته ایجاد نمی‌کنند و تنها می‌توان با آنها شروع عبارت را تشخیص داد بطوری که با دیدن صفر و یک‌های متناوب آن‌ها را دنبال می‌کنیم و اگر این تناوب تا ۸ بایت ادامه داشت (البته بجز بیت آخر که یک است.) به این معنی است که یک Ethernet frame آغاز شده است و ادامه مراحل را طی می‌کنیم.

در قسمت بعدی آدرس‌های MAC مربوط به مبدا و مقصد را دریافت می‌کنیم که با توجه به اینکه کاربردی ندارند تنها آن‌ها را ذخیره و نگهداری می‌کنیم و در آینده با آنها کاری نداریم. پس از آن دو بایت EtherType را دریافت می‌کنیم که براساس آنچه گفته شد فرض گرفته می‌شود مقداری کمتر از ۱۵۰۰ داشته و در نتیجه تعداد بایت‌های Payload را به ما نشان می‌دهد، بنابراین پس از دریافت این قسمت در قسمت بعدی با استفاده از مقدار EtherType تا تکمیل اطلاعات بیت ورودی را در Payload ذخیره می‌کنیم. دقت شود که ورودی بصورت سری به ما داده خواهد شد و بنابراین از یک رجیستر value برای بررسی آنکه هم اکنون در کجای رشته قرار داریم استفاده می‌شود. پس از دریافت این قسمت ۳۲ بیت یا همان ۴ بایت آخر را که مربوط به FCS است دریافت می‌کنیم. حال که تمامی اطلاعات مربوط به رشته را داریم باید بررسی صحت آن را انجام دهیم برای اینکار بجای قراردادن صفر در سمت راست مقدار Payload و انجام تقسیم مقدار FCS را قرار می‌دهیم. با توجه به اینکه در چند جمله‌ای اگر مقدارهای راست صفر بودند ما FCS را بدست می‌آوردیم در اینجا اگر خطایی در انتقال اطلاعات بوجود نیامده باشد باید خروجی ما بیت‌های متوالی صفر باشند. (دلیل این اتفاق آن است که در صورتی که اطلاعات سالم منتقل شده باشند دوباره مقدار FCS تولید می‌شود اما با توجه به آنکه تفریق مانند گیت XOR عمل می‌کند با مقدار FCS موجود در رشته اصلی XOR شده و تمامی بیت‌ها جداگانه به صفر تبدیل می‌شوند. اگر خروجی صفر باشد به معنای صحت انتقال اطلاعات از منظر CRC است و بنابراین Done بنابر با یک شده و مقادیر خواسته شده به خروجی می‌روند. در غیر اینصورت نیز Done صفر شده و تمامی مقادیر به مقدار اولیه خود باز می‌گردند تا آماده دریافت رشته بعدی شوند. یک نکته قابل ذکر دیگر آن است که در Ethernet بین فریم‌ها معمولا ۱۲ بایت (۹۶ بیت) فاصله وجود دارد که در اینجا با توجه به آنکه پیاده‌سازی ساختار دریافت خود فریم مدنظر بود از آن صرف نظر شده است.

توضیحات تکمیلی بوسیله کامنت‌گذاری در خود کد و تست‌بنچ مربوط به آن آورده شده است.

توضیحات درباره تست‌بنچ

باتوجه به اینکه تست‌بنچ دارای زمان اجرای ۴۶۴۰ نانوثانیه است که معادل ۴۶۴ دوره کلاک می‌باشد لذا توصیه می‌شود که ابتدا کد وریلاگ مطالعه و سپس شبیه‌ساز بررسی شود. تست‌بنچ دارای دو ورودی است که در ورودی اول اطلاعات بطور صحیح و بدون ایجاد اختلال انتقال یافته است. با توجه به این موضوع انتظار داریم در نانوثانیه ۲۳۲۰ شاهد یک پالس در Done باشیم و مقادیر مربوط به size و out مرتبط نیز در اینجا قابل بررسی هستند. (در واقع تا زمانی که مقدار Done برابر با صفر باشد یعنی ورودی صحیحی دریافت نکرده‌ایم و لذا خروجی‌ها ممکن است مقدار مورد نظر را نداشته باشند.)

در حالت دوم انتقال اطلاعات با اختلال مواجه شده است به این ترتیب خروجی به Done تبدیل نخواهد شد. پس از هربار دستیابی به خروجی Done تمامی مقدارها در کلاک بعدی ریست می‌شوند تا امکان دادن ورودی جدید فراهم شود. در حالت‌هایی که ورودی بطور کامل دریافت شده است اما Corrupted است (با توجه به اینکه این حالت از حالت‌هایی که هنوز ورودی بطور کامل دریافت نشده یا در حال دادن ورودی نیستیم متمایز است.) می‌توانیم با استفاده از دستورات دیگر از سیستم فرستنده درخواست ارسال دوباره اطلاعات کنیم که در واقع همان کاری است که در انتقال Packet های Ethernet در واقعیت انجام می‌گیرد. با توجه به هزینه کمتر ارسال دوباره اطلاعات در شبکه محلی نسبت به انتقال اطلاعات بیشتر برای ایجاد توانایی تصحیح، تنها تشخیص وجود یا عدم وجود خطا در داده انتقال یافته کافی است.

با تشکر از زحمات شما

امیرمهدی سلیمانی‌فر