به نام خدا

روش های کشف و اصلاح خطا در شبکه های کامپیوتری

علی نسیمی راد
۹۰۶۳۳۱۰۲۸

www.Prozhe.com

خطا در خطوط انتقال جزو حقایقی است که به هیچ وجه نمـی تـوان بطـور کامـل آن را برطـرف کـرد و همیشه جزو مشکلات عمدهٔ سیستم های مخابراتی بوده است. باید به طریقـی گیرنـده متوجـه دریافـت خطادار داده ها شود تا از آنها استفاده نکند.

کنترل خطا در شبکه های کامپیوتری به دو دسته کشف خطا و تصحیح خطا تقسیم مـی شـود،که هـردو اطلاعات افزونگی به داده های در حال ارسال اضافه می کنند.

در روش کشف خطا، آنقدر اطلاعات افزونگی به داده اصلی اضافه می شود که گیرنده از وقـوع یـا عـدم وقوع خطا آگاهی یابد و در صورت لزوم تکـرار ارسـال را خواسـتار شـود، ولـی در روش تصـحیح خطـا، اطلاعات افزونگی کافی به داده اصلی اضافه می شود به طوریکه گیرنـده بتوانـد داده واقعـی را از آنچـه دریافت کرده، استنتاج کند.

در مکانیزم کشف خطا، گیرنده متوجه وقوع خطا می شود ولی نمی تواند آن خطا را تصحیح کند. پس باید از فرستنده درخواست کند تا آن اطلاعات را دوباره ارسال کند. (ارسالNACK)

دریافت تصدیق مثبت (ACK) توسط فرستنده به معنی دریافت صحیح اطلاعات توسط گیرنده می باشد ولی دریافت تصدیق منفی(NACK) توسط فرستنده نشان می دهد که اوضاع رو به راه نیست و فریم باید مجددا ارسال شود.

در مکانیزم تصحیح خطا گیرنده علاوه بر کشف خطا می تواند خطای رخ داده را تصحیح کند.

مکانیزم های کشف / تصحیح خطا توسط لایه پیوند داده بر روی فریم انجام می شوند. در هنگام انتقـال داده در رسانه انتقال و در صورت رخداد خطا در فریم و با دریافت آن فریم توسط گیرنده لایه پیونـد داده گیرنده متوجه وقوع خطا می شود. یا آن را تصحیح می کند و یا درخواست ارسال مجدد مـی کنـد. البتـه تمامی این مراحل خطایابی از دید لایه بالاتر پنهان است.

در شبکه های کامپیوتری به علت سربار زیاد از مکانیزم های تصحیح خطا استفاده نمی شود مگر جاهایی که کانال از نوع کاملاً یک طرفه (Simplex) باشد. مثلاً به هنگام انتقال داده برای دستگاه های ریسـیور ماهواره به علت ذات کانال کاملاً یک طرفه از مکانیزم های تصحیح خطا استفاده می شود. در کانالهـای بسیار قابل اطمینان، مانند فیبرنوری، بهتر است از کد کشـف خطـا (ارسـال مجـدد فـریم هـای معیـوب) استفاده کنیم، ولی در یک کانال پرنویز (مثل لینک های بی سیم) که آهنگ بروز خطا بـالا اسـت، بهتـر است با اضافه کردن اطلاعات افزونگی بیشتر تصحیح خطا را به عهده گیرنده بگذاریم. در اکثر شبکه هـا از مکانیزم کشف خطا و درخواست ارسال مجدد اطلاعات استفاده می شود.

ماهیت خطا و علل بوجود آمدن آن را می توان در موارد زیر خلاصه کرد:

♦ **نویز حرارتی:** این نویز به دلیل حرکت اتفاقی الکترونها بوجود می آید و با افزایش دما، شـدت ایـن نویز هم به صورت خطی تقویت می شود . بخصوص در مداراتی مثل تقویت کننده های نیمه هـادی بـا ضریب تقویت و بهرهٔ بالا، تاثیر این نویز حساسیت بیشتری دارد . اثر این خطا کاملا تصادفی است.

www.Prozhe.com

♦ **شوکهای الکتریکی:** این نوع از نویز بدلیل قطع و وصل کلیدها، سیم ها و سوئیچ های الکتریکی یا رعد و برق بوجود آمده و نوعی خطای انفجاری را باعث می شود؛ یعنی مجموعهٔ گسترده ای از بیت ها که روی کانال در جریانند، به یکباره خراب می شوند. به عنوان مثال اگر یک شوک الکتریکی به اندازه 10msادامه یابد و اطلاعات روی کانال با سرعت 1Mbps در جریان باشد، با فرض آنکه طول متوسط فریم ها 1KB در نظر گرفته شود، این شوک می تواند تا ده فریم را بطور کلی نابود کند؛ به این معنا که فرستنده ده فریم را فرستاده ولی گیرنده هیچ فریمی دریافت نکرده است.

♦ **نویز کیهانی:** این نوع خطاها ناشی از حرکات کیهانی، کهکشان ها، وضعیت ستارگان و خورشید و امثال آن می باشد و تاثیر آن بیشتر بر روی کانال های رادیوئی است.

روش کشف خطا:

ساده ترین روش کشف خطا، اضافه کردن بیت توازن به داده ها است. در این روش به ازای هر بایت از اطلاعات یک بیت توازن اضافه می شود؛ این بیت باید به گونه ای انتخاب و اضافه شود که مجموع تعداد بیت های1 ، همیشه زوج یا فرد باشد.

مثال:

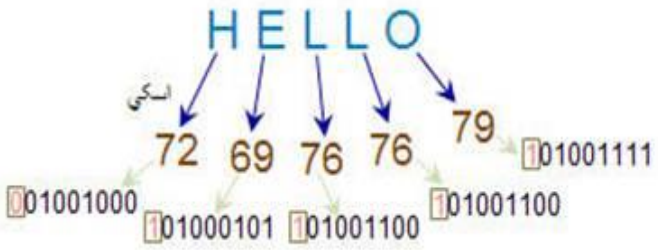
اصلی بایت : 01101001

توان بیت فرد 01101001 Odd Parity 1

توان بیت زوج 01101001 Even Parity 0

بنابراین گیرنده می تواند با بررسی بیت توازن، خطای احتمالی را کشف کند، ولی این روش در صورتی موثر است که تعداد خطاهای رخ داده زوج نباشد.

گیرنده می تواند با بررسی بیت توازن خطاهای احتمالی را تشخیص دهد، ولی این روش در صورتی موثر است که تعداد خطاهای رخ داده زوج نباشد. امکان تشخیص خطا برای یک خطا و خطاهای فرد(۳خطا، ۵ خطا و ...) وجود دارد.



الف) محاسبه توازن بیتها

۱- توازن بیتهایی را محاسبه می‌کنیم که باقیمانده تقسیم شماره بیت آنها بر ۲، یک باشد .

۲- توازن بیتهایی را محاسبه می‌کنیم که باقیمانده تقسیم شماره بیت آنها بر ۴، دو یا سه باشد .

۳- توازن بیتهایی را محاسبه می‌کنیم که باقیمانده تقسیم شماره بیت آنها بر ۸، چهار، پنج، شش یا هفت باشد .

۴- توازن بیتهایی را محاسبه می‌کنیم که باقیمانده تقسیم شماره بیت آنها بر ۱۶، هشت، نه، ده، یازده، دوازده، سیزده، چهارده یا پانزده باشد. والی آخر .

ب) سپس توازنهای به دست آمده را به ترتیب به صورت یک عدد دودویی می‌نویسیم. ارزش توازن به دست آمده در مرحله اول ۲۰، مرحله دوم ۲۱، مرحله سوم ۲۲ و الی آخر می‌باشد .

پ) اگر عدد به دست آمده در مرحله قبل صفر باشد یعنی خطایی وجود ندارد ولی اگر غیر صفر باشد، عدد حاصل شماره بیتی که خطا دارد را به صورت دودویی نشان می‌دهد. باید بیت مورد نظر معکوس شود .

ت) در مرحله آخر بیتهای کنترلی را حذف می‌کنیم و اگر در مرحله دو از Coding بیتی اضافه شده آن را نیز حذف میکنیم. در نهایت داده اصلی به دست می‌آید .

مثال :

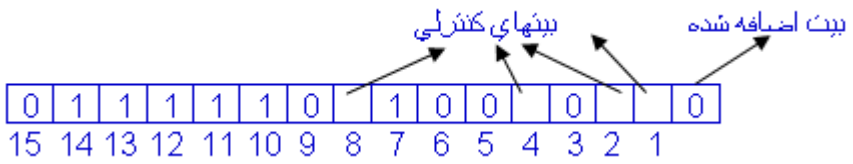
Coding:

داده اصلي 01111101000

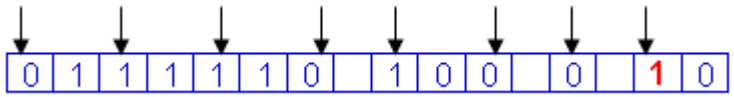
k= 11

m= 4

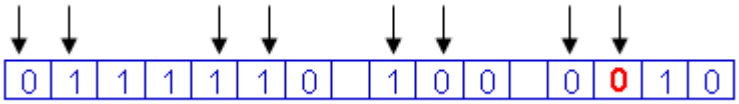
n=15



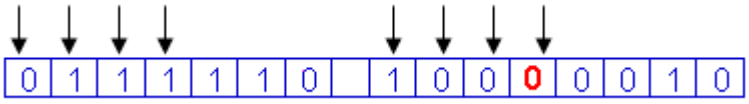
توازن بیتهایی را چک می‌کنیم که باقیمانده تقسیم شماره بیت آنها بر ۲، یک باشد



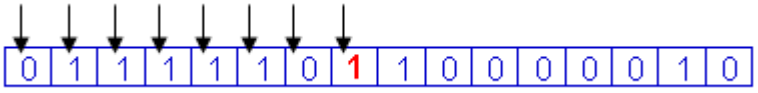
توازن بیتهایی را چک می‌کنیم که باقیمانده تقسیم شماره بیت آنها بر ۴، دو یا سه باشد



بیتهایی را چک می‌کنیم که باقیمانده تقسیم شماره بیت آنها بر ۸، چهار، پنج، شش یا هفت باشد



توازن بیتهایی را چک می‌کنیم که باقیمانده تقسیم شماره بیت آنها بر ۱۶، هشت، نه، ده، یازده، دوازده، سیزده، چهارده یا پانزده باشد.

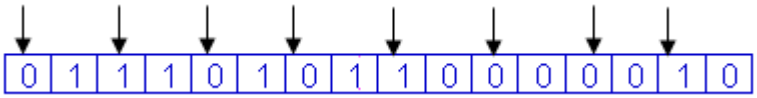


حال داده را با یک خطا در بیت یازدهم ارسال می‌کنیم. داده ارسال شده به صورت زیر خواهد بود :

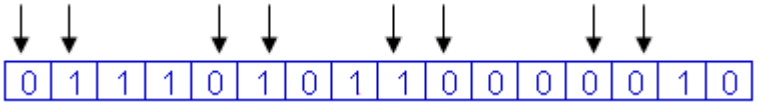
۰۱۱۱۰۱۰۱۱۰۰۰۰۰۱۰

Decoding:

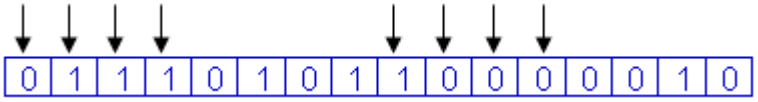
توازن بیتهایی را محاسبه می‌کنیم که باقیمانده تقسیم شماره بیت آنها بر ۲، یک باشد:P1=1



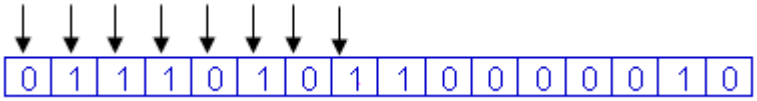
توازن بیتهایی را محاسبه می‌کنیم که باقیمانده تقسیم شماره بیت آنها بر ۴، دو یا سه باشد:P2=1



توازن بیتهایی را محاسبه می‌کنیم که باقیمانده تقسیم شماره بیت آنها بر ۸، چهار، پنج، شش یا هفت باشد: P3=0



توازن بیتهایی را محاسبه می‌کنیم که باقیمانده تقسیم شماره بیت آنها بر شانزده، هشت، نه، ده، یازده، دوازده، سیزده، چهارده یا پانزده باشد:P4=1

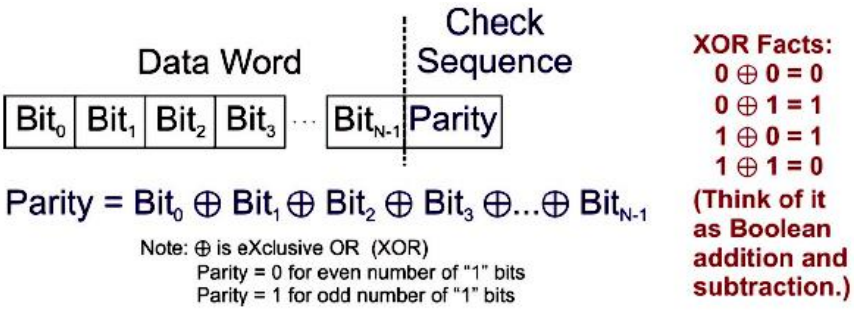


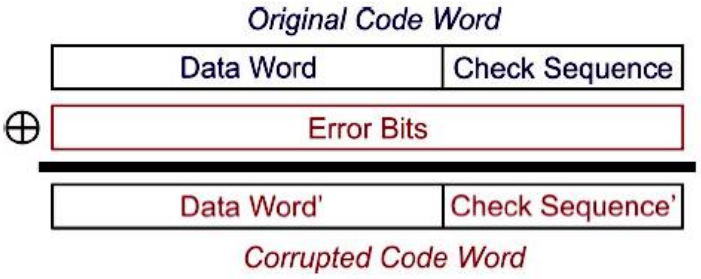
توازنهای به دست آمده نشان می‌دهد که بیت یازدهم دارای خطا است پس طبق الگوریتم بیت یازدهم را معکوس می‌کنیم در نتیجه خواهیم داشت:

۰۱۱۱۱۱۰۱۱۰۰۰۰۰۱۰

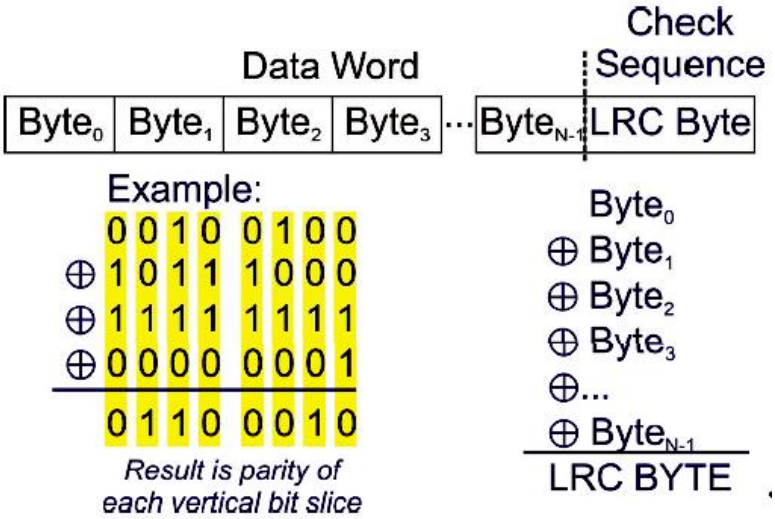
پس از حذف بیتهای کنترلی و بیت اضافه شده داده اصلی به دست خواهد آمد.

۰۱۱۱۱۱۰۱۰۰۰



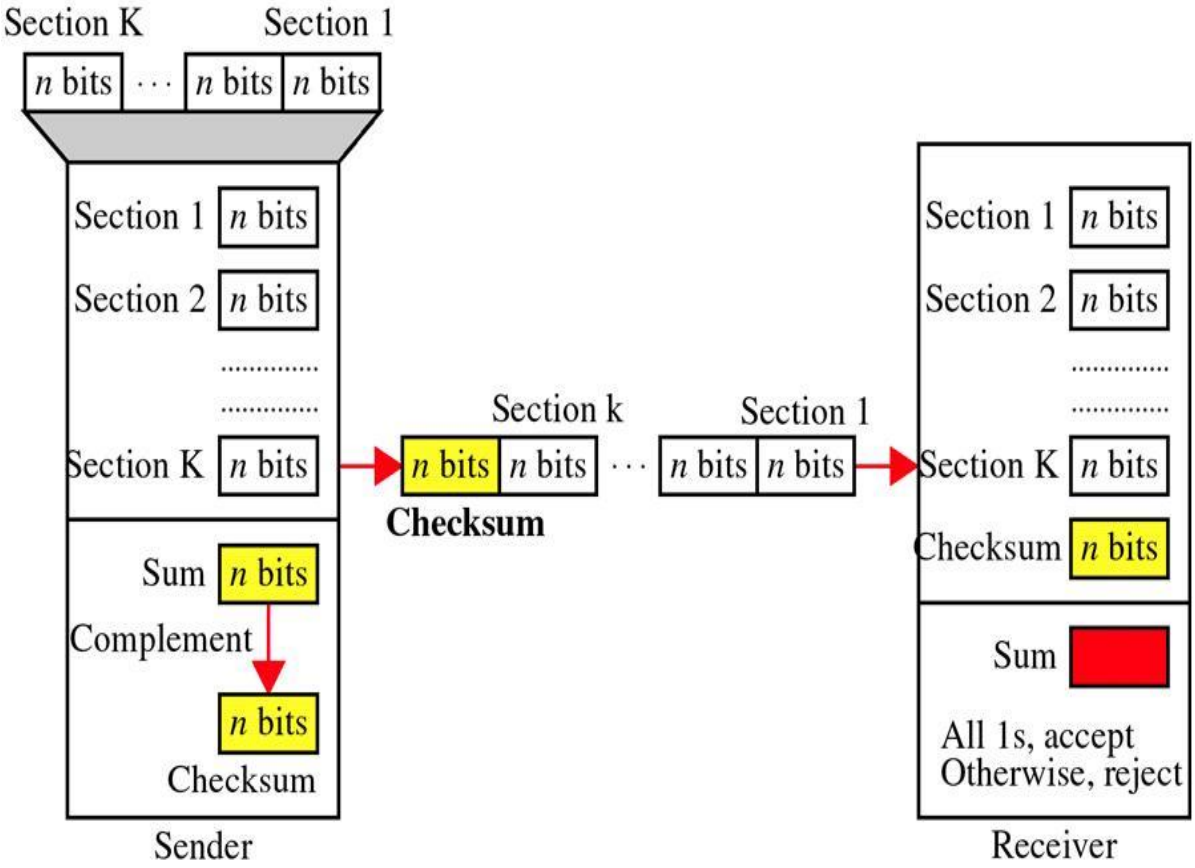


Error detection works if the corrupted Code Word is invalid



روش Checksum **:**

در این روش تمام بایت های یک فریم که باید توسط فرستنده ارسال شود، با هم جمع (XOR) شده و یک بایت به نام Checksum بدست می آید. این بایت در انتهای فریم به مقصد ارسال می شود. در مقصد مجدداً بایت Checksum محاسبه و سپس مقایسه می شود. این روش در صورتی قادر به کشف خطا است که تعداد خطاهای رخ داده در بیت های هم ارزش زوج نباشد.



برخی از الگوریتمهای چک سام:

برخی از الگوریتم های ساده چک سام ، الگوریتم ۸ – ۱۶ – ۳۲ – ۶۴ بیتی و CRC16 - CRC32 بیتی و ... می باشند. دقت کنید که بین الگوریتم چک سام و الگوریتم CRC تفاوت هایی هست. چک سام براساس هر الگوریتمی که باشد تعداد بیتهای (Bit) آن تقسیم بر ۸ برابر با تعداد بایت (Byte) می شود. مثلاً اگر چک سام بر اساس الگوریتم Checksum8 بیتی باشد مقدار چک سام ما ۱ بایت است. اگر چک سام بر اساس الگوریتم Checksum16 باشد مقدار چک سام ما ۲ بایت است. و اگر چک سام بر اساس الگوریتم Checksum32 باشد مقدار چک سام ما ۴ بایت است ... الی آخر...

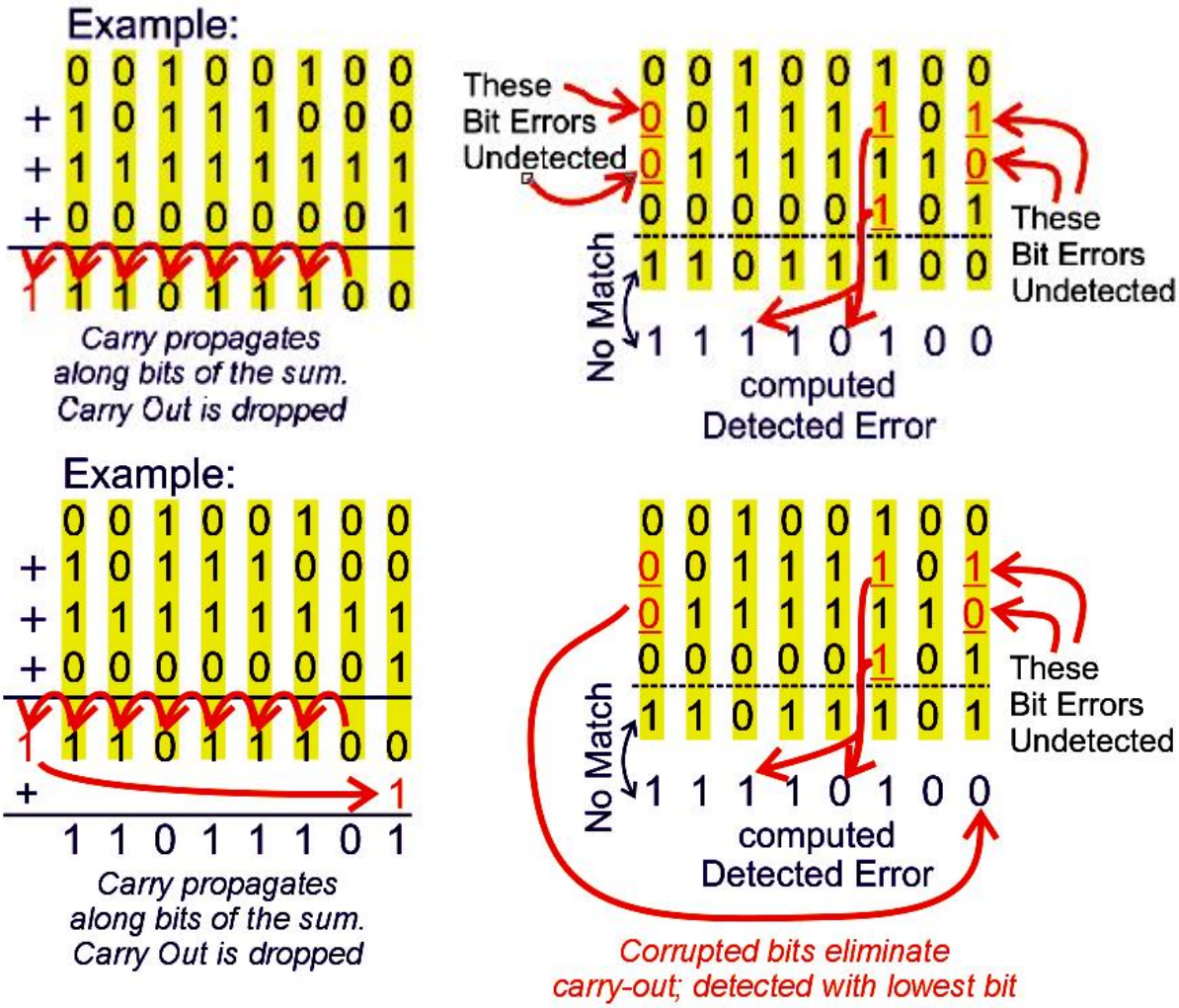
بطور کلی هر بیت از فایل دارای چک سام خود است که بر اساس الگوریتمی استاندارد و مطلق محاسبه می شود. از سوی دگر اگر در یک فایل چک سام ما بر اساس الگوریتم CRC16 بیتی باشد، از آنجاییکه

هر ۸ بیت برابر با یک بایت می شود پس چک سام ما دو بایت از فایل را شامل می شود. الگـوریتم هـا برای محاسبه چک سام بر اساس محاسبات ریاضی و بصورت استاندارد و در دسترس می باشند.

محاسبه چک سام:

برای محاسبه چک سام یک فایل ، نخست باید بدانیم که چک سام ما بر اساس چه الگوریتمی محاسبه شده است سپس بایستی مکان قرار گیری چک سام را شناسایی کنیم. و در آخر هم باید بدانیم که آیا با " Ignore Bytes "هم طرف هستیم یا نه و اگر بله مکان آن کجاست ؟

اکنون در زمان تغییرات معقول یک فایل می توانیم بـه سـادگی چـک سـام تـازه آن را هـم محاسـبه و جایگزین کنیم. گاهی وقت ها هم در ویرایش یک فایل برای تعدیل مقدار چـک سـام، بایـت هـایی بـه انتهای فایل اضافه می کنند که البته در این راه هم به دانستن الگوریتم چک سام در آن فایل و هـم بـه اطلاعات پیش زمینه دیگری نیاز داریم.



کدهای کشف خطای CRC (Cyclic Redundancy Check):

در روش CRC، به ازای مجموعه ای از بیت ها (مثلاً ۵۱۲۰ بیت یا ۱۰۲۴۰ بیت ...) تعدادی بیت کنترلی به نام CRC محاسبه و به انتهای فریم اضافه می شود. مبنای محاسبهٔ کدهای CRC با استفاده از تقسیم چندجمله ای است که روش محاسبهٔ آن با یک مثال توضیح داده شده است:

دادهٔ اصلی: 11100101

7	6	5	4	3	2	1	0
1	1	1	0	0	1	0	1

ابتدا از روی دادهٔ اصلی یک چندجمله ای تولید می شود. نمایش ریاضی چند جمله ای بدینصورت است که بیت ها از راست به چپ ضرایب یک چند جمله ای قرار می گیرند که توان هر جمله را موقعیت بیت در رشته مشخص می کند. بدین صورت داده به صورت یک چند جمله ای نمایش داده خواهد شد. رشتهٔ بیت در این مثال برای سادگی عملیات ، هشت بیتی فرض شده است ، ولی در عمل این رشته می تواند ده ها هزار بیت طول داشته باشد.

$D(X) = 1 * x^7 + 1 * x^6 + 1 * x^5 + 0 * x^4 + 0 * x^3 + 1 * x^2 + 0 * x + 1$

$D(X) = x^7 + x^6 + x^5 + x^2 + 1$

برای تولید کد CRC ، چند جمله ای $x^n.D(X)$ بر یک" چندجملهای مولد " که بین گیرنده و فرستنده توافق می شود و اختیاری است ، تقسیم می گردد. (n بالاترین توان چندجمله ای مولد است.) تقسیم در مبنای 2 انجام می شود ، یعنی ضرایب جملات با توان مساوی با هم XOR خواهد شد و تفریق معنا ندارد. باقیماندهٔ تقسیم به عنوان کدهای کنترل خطا در انتهای داده ها ارسال خواهند شد.

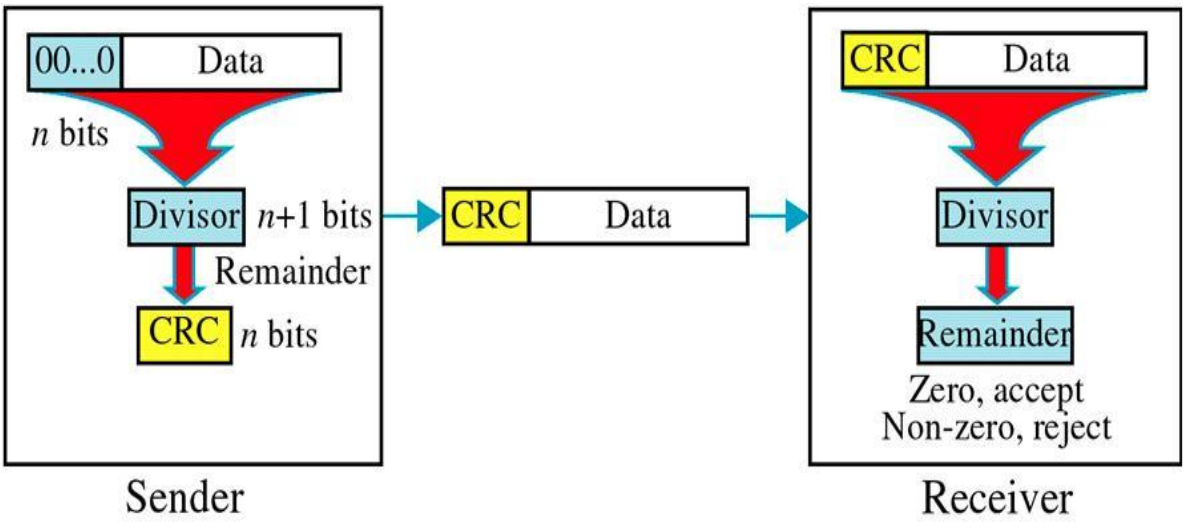
CRCمولد = $X^2 + 1 \rightarrow 101$

Data = $x^7 + x^6 + x^5 + x^2 + 1 * X^2 \xrightarrow{*x^2} X^9 + X^8 + X^7 + X^4 + X^2$

$x^9 + x^8 + x^7 + x^4 + x^2 / x^2 + 1$

باقیمانده : +1 = 01

در روشی که بخواهیم بصورت باینری (به جای چند جمله ای) کد CRC را حساب کنیم ، به تعداد بزرگترین توان جملهٔ مولد ، در سمت راست رشتهٔ داده صفر اضافه می کنیم. بنابراین در مثال بالا باید دو صفر به سمت راستِ داده اضافه شود. سپس داده ای که به آن صفر اضافه شده است ، بر چند جمله ای مولد تقسیم می شود.



در تقسیم به نکات زیر باید توجه داشت:

۱- تقسیم از این لحاظ با تقسیم معمولی متفاوت است که شما باید از سمت چپ بیت های باقیمانده را صفر کنید.

۲- جمع در مبنای۲ و به صورت XOR انجام می شود.

۳- نهایتاً بیت های باقیماندهٔ تقسیم در سمت راست بیت های داده قرار می گیرد . مثال:

```
1 1 1 0 0 1 0 1 0 0   | 101
1 0 1                 11010001
  1 0 0
  1 0 1
      1 0 1
      1 0 1
        0 0 0 1 0 0
            1 0 1
              0 1 کد CRC
```

```
Frame:      1 1 0 1 0 1 1 1 1 1
Generator:  1 0 0 1 1

                    1 1 0 0 0 0 1 1 1 0  ← Quotient (thrown away)
1 0 0 1 1 / 1 1 0 1 0 1 1 1 1 1 0 0 0 0  ← Frame with four zeros appended
            1 0 0 1 1
              1 0 0 1 1
              1 0 0 1 1
                0 0 0 0 1
                0 0 0 0 0
                  0 0 0 1 1
                  0 0 0 0 0
                    0 0 1 1 1
                    0 0 0 0 0
                      0 1 1 1 1
                      0 0 0 0 0
                        1 1 1 1 0
                        1 0 0 1 1
                          1 1 0 1 0
                          1 0 0 1 1
                            1 0 0 1 0
                            1 0 0 1 1
                              0 0 0 1 0
                              0 0 0 0 0
                                    1 0  ← Remainder

Transmitted frame:  1 1 0 1 0 1 1 1 1 1 0 0 1 0  ← Frame with four zeros appended
                                                   minus remainder
```

مثال:

Data = $X^6 + X^2 + 1$

CRC Generator = $X^2 + 1$

```
100010100 | 101
101
00101
  101
  0000100
      101
```

CRC کد : 01

مثالی از مولدهای شناخته شده و استاندارد CRC :

CRC-12 = $X^{12} + X^{11} + X^3 + X^2 + 1$

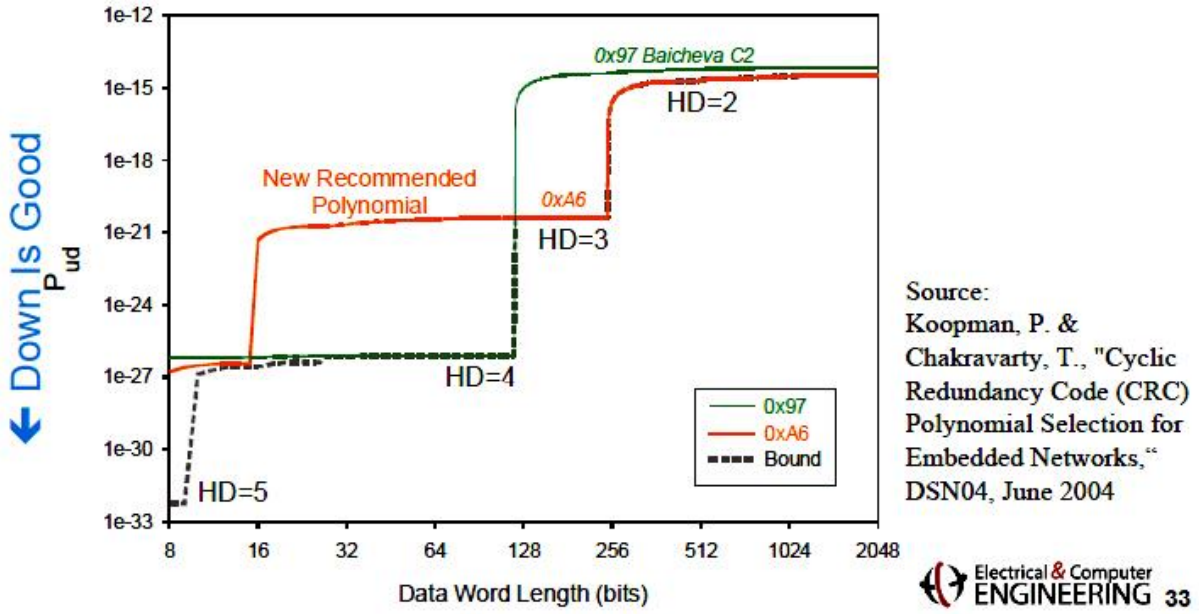
CRC-16 = $X^{16} + X^{15} + X^2 + 1$

CRC-CCITT = $X^{16} + X^{12} + X^5 + 1$

کدهای محاسبه شدهٔ CRC معمولا در انتهای اطلاعات ارسال خواهد شد و پس از دریافت اطلاعـات در گیرنده ، مجدداً کدهای CRC برای داده ها محاسبه می شوند و نتیجه با کد ارسـالی CRC مقایسـه مـی گردد و در صورت عدم تطابق، خطایی در داده ها وجود دارد و داده ها فاقد اعتبار است. اگـر مولـد CRC مناسب انتخاب شود، احتمال آنکه خطایی بروز کند ولی گیرنده قادر به کشف آن نباشد، کمتـر از 0.002 است.

دقت کنید که عمل محاسبهٔ کدهای CRC و همچنین بررسی خطا از طریق تراشه های سـخت افـزاری انجام می شود تا سرعت عمل بالا برود. این تراشه ها بسادگی و از طریق شیفت رجیسترهای فیدبک دار و گیت های منطقی ساده مثل XOR پیاده می شود.

الگوریتم هایی مانند Alpha Blending ، Rendering ، Transparency و غیره تا قسمت عمـده ای بر پایه XOR بنا شده اند.



Max length at HD	CRC Size (bits)													
Polynomial	3	4	5	6	7	8	9	10	11	12	13	14	15	16
HD=2	2048+ 0x5	2048+ 0x9	2048+ 0x12	2048+ 0x21	2048+ 0x48	2048+ 0xA6	2048+ 0x167	2048+ 0x327	2048+ 0x64D	–	–	–	–	–
HD=3		11 0x9	26 0x12	57 0x21	120 0x48	247 0xA6	502 0x167	1013 0x327	2036 0x64D	2048 0xB75	–	–	–	–
HD=4			10 0x15	25 0x2C	56 0x5B	119 0x97	246 0x14B	501 0x319	1012 0x583	2035 0xC07	2048 0x102A	2048 0x21E8	2048 0x4976	2048 0xBAAD
HD=5						9 0x9C	13 0x185	21 0x2B9	25 0x5D7	53 0x8F8	none	113 0x212D	136 0x6A8D	241 0xAC9A
HD=6							8 0x13C	12 0x28E	22 0x532	27 0xB41	52 0x1909	57 0x372B	114 0x573A	135 0xC86C
HD=7									12 0x571	none	12 0x12A5	13 0x28A9	16 0x5BD5	19 0x968B
HD=8										11 0xA4F	11 0x10B7	11 0x2371	12 0x630B	15 0x8FDB