گزارشِ تحقیق درباره‌ی الگوریتم‌های مختلف تصحیحِ خطا

تهیه و تنظیم: مبین خیبری

شماره دانشجویی: 994421017

استاد راهنما: دکتر میرسامان تاجبخش

چکیده:

در گزارشِ پیش‌رو قصد داریم روش‌های مختلفِ تصحیح خطاهای بیتی را با تشریحِ نحوه‌ی محاسبه و تصحیحِ خطا، معرفی کرده و به مقایسه‌ی عملکردِ آن‌ها با یکدیگر بپردازیم. همچنین برای درک بهتر الگوریتم‌های مختلف موجود جهتِ تشخیص و اصلاحِ خطا در شبکه‌های کامپیوتری، نگاهی به تاریخچه‌ی ظهور و تکامل این روش‌ها نیز خواهیم انداخت.

شناسایی و اصلاح خطا در شبکه های کامپیوتری

عوامل زیادی همچون نویز، تداخل خطوط و غیره وجود دارند که می‌توانند باعث از بین رفتن داده‌ها در طی انتقال شوند. لایه‌های بالاتر در نمایی تعمیم یافته از معماری شبکه قرار دارند و از روش پردازش داده‌ها روی شبکه واقعی اطلاع ندارند. از این رو لایه‌های فوقانی انتظار یک انتقال عاری از خطا بین سیستم‌ها را دارند. اغلب اپلیکیشن‌ها در صورت وجود داده‌های خطادار مطابق انتظار رفتار نخواهند کرد. با این وجود، کاربردهایی مانند انتقال صوت و تصویر ممکن است تا این حد متأثر از خطاها نباشند و در صورت وجود پاره‌ای خطاها، همچنان به درستی کار کنند.

لایه داده-لینک از نوعی مکانیسم کنترل خطا استفاده می‌کند تا اطمینان یابد که قاب‌ها (Frames) (جریان‌های بیتی داده) با سطح معینی از دقت انتقال می‌یابند. اما برای درک چگونگی کنترل خطا می‌بایست با انواع خطاهایی که ممکن است پیش بیایند آشنا باشیم.

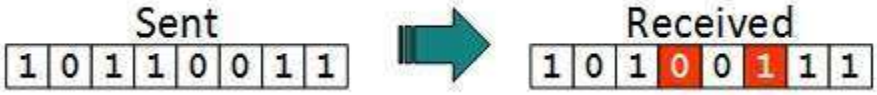
انواع خطاها

سه نوع خطا ممکن است در شبکه‌های کامپیوتری رخ دهد:

خطای یک بیت منفرد

در یک قاب، تنها یک بیت وجود دارد که به نوعی از بین رفته است.

خطای چند بیت



فریم در حالی دریافت می‌شود که بیش از یک بیت از بین رفته است.

خطای گسترده

فریم شامل چندین بیت متوالی از داده‌های از بین رفته است.

مکانیسم کنترل خطا به دو صورت می‌تواند عمل کند که عبارت هستند از شناسایی خطا و اصلاح خطا و در ادامه به آن‌ها خواهیم پرداخت.

شناسایی خطا

خطاها در فریم‌های دریافتی به وسیله «بررسی توازن» (Parity Check) و بررسی «افزونگی چرخه‌ای» (Cyclic Redundancy) شناسایی می‌شوند. در هر دو حالت، چند بیت اضافی همراه با داده‌های واقعی ارسال می‌شوند تا تأیید شود که بیت‌های دریافتی در سمت دیگر همان‌هایی هستند که ارسال شده‌اند. اگر بررسی متقابل در سمت گیرنده با شکست مواجه شود، بیت‌ها به صورت از بین رفته تلقی می‌شوند.

بررسی توازن

یک بیت اضافی همراه با بیت‌های اصلی ارسال می‌شود تا در صورتی که توازن زوج وجود دارد، تعداد 1-ها زوج شود و یا در صورت وجود توازن فرد، تعداد -1ها فرد شود.

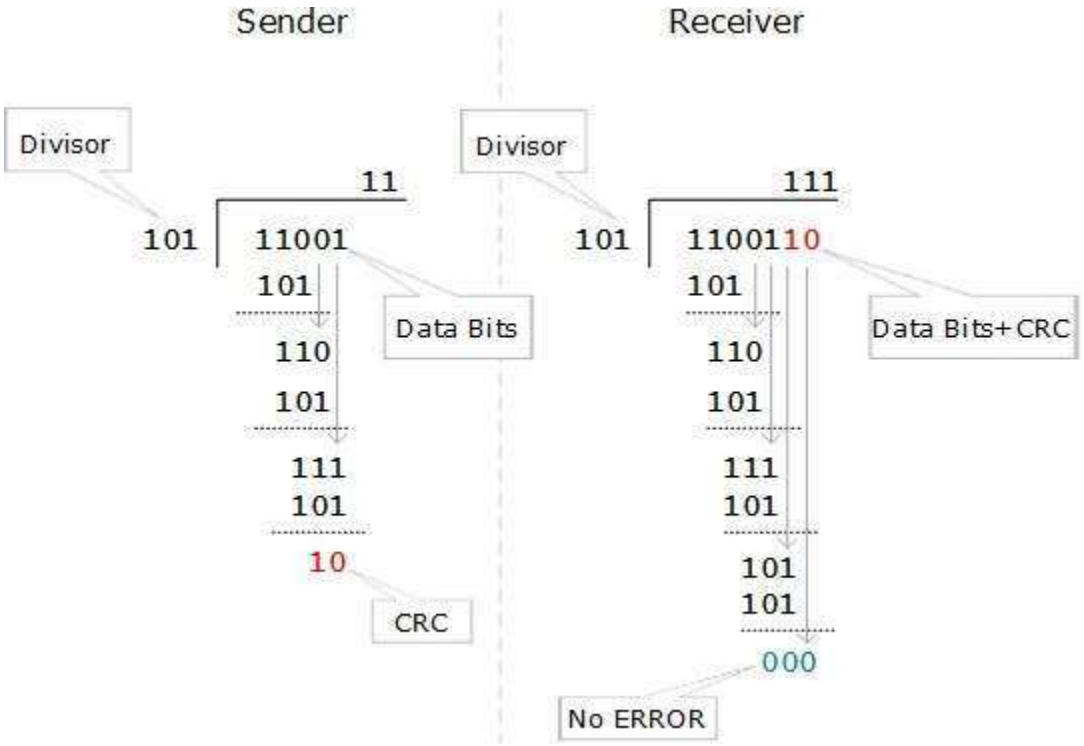
در این روش فرستنده در زمان ایجاد یک فریم تعداد 1-های داخل آن را می‌شمارد. برای نمونه اگر از توازن زوج استفاده شود و تعداد 1-ها زوج باشد، یک بیت با مقدار 0 اضافه می‌شود. بدین ترتیب تعداد 1-ها زوج باقی می‌ماند .اگر تعداد 1-ها فرد باشد، برای این که زوج شود، یک مقدار 1 دیگر اضافه می‌شود.

در این روش، سمت گیرنده تعداد 1-ها در فریم را می‌شمارد. گر تعداد 1-ها زوج باشد و توازن زوج برقرار باشد، قاب به صورت سالم تصور می‌شود و مورد پذیرش قرار می‌گیرد. اگر تعداد 1-ها فرد باشد و توازن فرد برقرار باشد، همچنان فریم سالم محسوب می‌شود.

اگر در زمان انتقال، تنها یک بیت معکوس شده باشد، گیرنده می‌تواند با شماره تعداد 1-ها این وضعیت را تشخیص دهد. اما زمانی که بیش از یک بیت دارای خطا باشد، در این صورت شناسایی خطا برای گیرنده دشوار خواهد بود.

بررسی افزونگی چرخه‌ای(CRC)

CRC رویکردی متفاوت برای شناسایی سالم بودن داده‌های دریافتی است. این تکنیک شامل تقسیم باینری بیت‌های داده ارسالی است. مقسوم با استفاده از معادله‌های چندجمله‌ای تشکیل می‌شود. در این روش فرستنده عملیات تقسیم را روی بیت‌هایی که قرار است ارسال شوند، انجام داده و باقی‌مانده را محاسبه می‌کند. پیش از ارسال کردن بیت‌های واقعی، فرستنده باقی‌مانده را به انتهای بیت‌های واقعی اضافه می‌کند. بیت‌های داده واقعی به علاوه باقیمانده به نام «کلمه رمز» (Codeword) شناخته می‌شوند. فرستنده بیت‌های داده را به صوت کلمه رمز ارسال می‌کند.



در سمت دیگر، گیرنده عملیات تقسیم را با استفاده از همان مقسوم روی کلمه رمز اجرا می‌کند. اگر باقیمانده کلاً برابر با بیت‌های صفر باشد، داده‌های دریافتی مورد پذیرش قرار می‌گیرند؛ در غیر این صورت داده‌های دریافتی به صورت نوعی داده از بین رفته در زمان انتقال محسوب می‌شوند.

اصلاح خطا

در دنیای دیجیتال، اصلاح خطا به دو روش صورت می‌گیرد:

اصلاح خطای رو به عقب

هنگامی که گیرنده خطایی را در داده‌های دریافتی تشخیص دهد، از فرستنده تقاضا می‌کند که داده‌ها را یک بار دیگر ارسال کند.

اصلاح خطای رو به جلو

زمانی که گیرنده نوعی خطا را در داده‌های دریافتی شناسایی کند، کد اصلاح خطایی را اجرا می‌کند که به بازیابی خودکار و اصلاح برخی از انواع خطا کمک می‌کند.

روش اول که اصلاح خطای رو به عقب نام دارد، آسان است و تنها در مواردی به صورت مؤثر قابل اجرا است که ارسال مجدد داده‌ها مستلزم هزینه بالایی نباشد. برای نمونه فیبر نوری چنین است. اما در مورد روش انتقال بی‌سیم، ارسال مجدد ممکن است هزینه بالایی داشته باشد. در این موارد اخیر، از روش اصلاح رو به جلو استفاده می‌شود.

برای اصلاح خطا در فریم داده‌ها، گیرنده باید دقیقاً بداند که کدام بیت در فریم از بین رفته است. برای موقعیت‌یابی خطا، بیت‌های تکراری به عنوان بیت توازن برای شناسایی خطا مورد استفاده قرار می‌گیرند. برای نمونه یک «کلمه» (Word) یا 7 بیت داده را در قالب ASCII دریافت می‌کنیم و سپس می‌توانیم 8 نوع اطلاعات مورد نیاز خود را داشته باشیم که 7 بیت به ما می‌گویند کدام بیت خطا دارد و یک بیت دیگر به ما اعلام می‌کند که خطایی وجود ندارد.

برای m بیت داده، rبیت افزونگی مورد استفاده قرار می‌گیرد. این r بیت می‌توانند، تعداد 2 rترکیب اطلاعات ارائه کنند. در یک کلمه رمز m+r بیتی، این احتمال وجود دارد که r بیت خودشان نیز از بین بروند. بنابراین تعداد r بیت استفاده می‌شود تا مطمئن شویم که موقعیت m+r بیت کجاست و همچنین اطلاعات بدون خطا هستند، یعنی m+r+1 مورد نیاز هستند.

$$2^r >= m+r+1$$

تشخیص و تصحیح خطا

تشخیص و تصحیح خطا (به انگلیسی: Error detection and correction) در نظریه اطلاعات، نظریه کدگذاری، علوم رایانه، مخابرات تکنیک‌هایی هستند که تحویل امن داده‌ها در کانال‌های مخابراتی ناامن را ممکن می‌کنند بسیاری از کانال‌ها در معرض نویز هستند و ممکن است اطلاعات در حین فرستاده‌شدن میان مبدأ و مقصد دچار خطا گردند. تشخیص و تصحیح خطا امکان شناسایی و ساخت مجدد اطلاعات اولیه را ممکن می‌گرداند.

تعریف

تشخیص خطا: تشخیص خطاهایی که با نویز یا با اختلال‌هایی ایجاد می‌گردند و درهنگام انتقال میان فرستنده و گیرنده به وجود می‌آیند.

تصحیح خطا: یافتن خطا و بازیابی اصل اطلاعات

تاریخچه

معروف‌ترین استفاده قاعده‌دار از تشخیص خطا توسط کاتبان یهودی در هنگام کپی‌برداری از کتاب مقدسشان بود آن‌ها روش‌های مختلفی داشتند از قبیل جمع کلمات در هر خط یا جمع کلمات در هر صفحه یا چک کردن یک پاراگراف (معمولاً پاراگراف میانی). در هر صفحه اگر حتی یک خطا وجود داشت کل صفحه ازبین می‌رفت، اما اگر ۳ خطا هم‌زمان دریک صفحه رخ می‌داد؛ کل صفحه رانابود می‌کردند. این روش کار بودن خود را زمانی نشان داد که طومارهای دریای مرده کشف شدند.

معرفی

ایده عمومی این است که چیزی به متن اصلی افزوده گردد که دریافت‌کننده بتواند درستی متن دریافتی را بررسی نماید، اگر خطایی محرز گشت آن را تصحیح کند. طرح‌های تشخیص و تصحیح خطا می‌تواند به صورت سیستماتیک یا غیرسیستماتیک باشد. در طرح سیستماتیک فرستنده داده‌های اصلی را همراه تعداد ثابتی عدد به عنوان بیت‌های بررسی می‌فرستد، که بیت‌های بررسی از الگوریتم قطعیای به دست می‌آیند که از داده‌های اصلی استفاده می‌کنند. اگر فقط تشخیص خطا مدنظر باشد گیرنده می‌تواند الگوریتم را دوباره بر روی داده‌های اصلی اجرا و مقدار خروجی آن را با بیت‌های بررسی مقایسه کند اگر یکسان بودند خطایی رخ نداده‌است. در سامانه‌هایی که از کد غیر سیستماتیک استفاده می‌کنند، پیام اصلی تبدیل به یک پیام کد شده می‌شود. عملکرد مناسب زمانی حاصل می‌گردند که بر اساس ویژگی‌های کانال مخابراتی و طرح‌های انتقال داده انتخاب گردند. انواع معمول کانال‌های مخابراتی شامل مدل بدون حافظه که در آن خطا به صورت تصادفی و با احتمال قطعی اتفاق می‌افتد و مدل‌های پویا است. در نتیجه تشخیص و تصحیح خطا را می‌توان به (به انگلیسی: random-error-detecting/correcting) و (به انگلیسی: burst-error-detecting/correcting) تقسیم کرد. اگر ظرفیت کانال را نتوان معین کرد یا ظرفیت بیش از حد متغیر باشد می‌توان درخواست ارسال مجدد داده‌ها را داشت که آن را به عنوان درخواست بازفرستی خودکار، (به انگلیسی: automatic repeat request) می‌شناسند که به طور ویژه در اینترنت کاربرد دارد.

اجرا

تشخیص و تصحیح خطا به دو صورت زیر تحقق میابد:

- درخواست بازفرستی خودکار، (به انگلیسی: automatic repeat request):داده‌ها به فرم بلوکی دریافت می‌گردند هر بلوک داده برای وجود خطا بررسی می‌شود اگر خطایی یافت شود، به صورت خودکار برای آن بلاک داده درخواست ارسال مجدد می‌شود این روند ادامه میابد تا زمانی که کل داده‌ها به صورت سالم دریافت گردند.

- اصلاح خطا رو به جلو (به انگلیسی: Forward error correction): فرستنده اطلاعات را قبل فرستادن با کمک (به انگلیسی: error-correcting code (ECC) کد می‌کند. اطلاعات اضافه شده توسط کد دریافت‌کننده برای بازیابی اطلاعات اصلی مورد استفاده قرار می‌گیرد.

این دو روش ممکن است ترکیب گردند وروش دیگری به نام درخواست تکرار اتوماتیک ترکیبی (به انگلیسی: Hybrid automatic repeat request) بسازند.

طرح‌های تشخیص خطا

تشخیص خطا اغلب با یک تابع درهم‌سازی مناسب صورت می‌گیرد یک تابع هش برچسب‌هایی باطول ثابت را به پیام می‌افزاید و که گیرنده را قادر می‌سازد که پیام دریافتی را با بررسی برچسب صحت آن را تأیید کند. طرح‌های بسیار زیادی برای تابع‌های هشی وجود دارد اما استفاده از آن‌ها بسته به سادگی یا مناسب بودن برای تشخیص نوع خاصی از خطاهست.

کدهای تکرار

اساس این طرح این است که کد ارسالی چندین بار ارسال گردد تا از صحت آن مطمئن شویم یک کد بزرگ به چندین بلاک تقسیم می‌گردد، آنگاه هر کدام چند بار مثلاً ۳ بار ارسال می‌گردند؛ بنابراین اگر یکی از این بلاک‌ها متفاوت با دیگری باشد خطایی رخ داده هست. این طرح کارا نیست و زمان زیادی می‌گیرد، اما مزیت آن سادگی آن است.

بیت توازن

دراین حالت یک بیت اضافه می‌شود به گونه‌ای که کل بیت‌های ارسالی دارای مجموع اعضای فرد یا زوج باشد.

چک‌سام

چک‌سام یک پیام عبارت است از هم‌نهشتی مجموع کلمات پیام.

کد افرونگی چرخشی

یک تابع درهم‌ساز غیر ایمن هست که برای تشخیص خطاهایی تصادفی در شبکه ایجاد شده‌است.

تابع درهمساز رمزنگارانه

تابع درهم‌سازی هست که رشته‌ای را می‌گیرد و رشته‌ای ثابت برمی‌گرداند این طرح اطمینان بالایی از لحاظ یکپارچگی داده دارد. هرگونه تغییر در داده باعث تغییر مقدار تابع درهم‌ساز شده و خطا یافته می‌شود.

اصلاح خطا روبه جلو

(به انگلیسی: Forward error correction)، با هر الگوی اصلاح خطا روبه جلو می‌توان خطاها را یافت. با کمترین فاصله همینگ d می‌تواندتا1-d خطا را یافت. این طرح مناسب است اگر بتوان حداقل تعداد خطاها را قبل از ارسال پیش‌بینی نمود.

تصحیح خطا

درخواست بازفرستی خودکار

یک روش کنترل خطا که از کدهای کشف خطا، قبول یا عدم قبول پیام و وقفه برای انتقال مطمئن داده‌ها استفاده می‌کند (قبول پیام یعنی گیرنده پیامی به فرستنده می‌فرستد که پیام به درستی دریافت شده‌است). فرستنده به‌طور معمول در زمان وقفه اگر پیام قبولی دریافت نکند، پیام را مجدداً می‌فرستد و این کار تا تعداد معینی بار صورت می‌پذیرد. این روش برای کانال‌های ارتباطی متغیر یا با ظرفیت ناشناخته مانند اینترنت مناسب هست.

اصلاح خطا روبه جلو

بر این اساس افزونه‌ای به کد اضافه گشته تا دریافت‌کننده بتواند آن را حتی با وجود تعداد معینی خطا بازیابی کند. اصلاح خطا رو به جلو در مدل اتصال متقابل سامانه‌های باز کاربرد دارد.

درخواست بازفرستی خودکار ترکیبی

از ترکیب دو روش قبل به دست می‌آید و دو روش اساسی دارد:

- پیام‌ها با روش اصلاح خطا روبه جلو فرستاده و بعد گیرنده پیام را کد گشایی می‌کند، اگر کد گشایی درست انجام نشود، مانند درخواست بازفرستی مجدد، درخواست بازفرستادن پیام را می‌کند.
- پیام فقط با اطلاعاتی دربارهٔ چگونگی تشخیص خطا فرستاده می‌شود، اگر گیرنده خطایی بیابد، با روش درخواست بازفرستی مجدد خودکار درخواست اطلاعاتی دربارهٔ چگونگی تصحیح آ نرا درخواست می‌کند تا بتواند اطلاعات را بازیابی کند.

موارد غیرقابل استفاده

برنامه‌هایی که احتیاج به زمان تأخیر کم دارند (مانند گفتگوی تلفنی) نمی‌توان از درخواست بازفرستی خودکار استفاده کرد و باید از اصلاح خطا رو به جلو استفاده کرد. همچنین مانند مورد قبل برای برنامه‌هایی که وقتی اطلاعات را می‌فرستند دیگران را دردسترس برای ارسال مجدد ندارند نمی‌توان از درخواست باز فرستی خودکار استفاده کرد.

بعضی موارد کاربرد

ارتباط تنگاتنگی میان توسعه اصلاح خطا رو به جلو و برنامه‌های فضایی بود. در فضا مسافت زیاد، قدرت سیگنال کم و نیروی مورد نیاز محدود است، پس احتیاج بود اطلاعات ارسالی اگر هم مشکل دارند در

همان‌جا تصحیح و مورد استفاده قرار گیرند. برای همین هم برای مثال وویجر ۲ از تصحیح خطای رید-سالامون استفاده کرده‌است. همچنین از تشخیص و اصلاح خطا در بالابردن قابلیت اطمینان فضاهای ذخیره‌سازی داده‌ها استفاده می‌گردد برای مثال در هارد درایوهای جدید از کد افزونه‌ای چرخشی برای تشخیص و از تصحیح خطای رید-سالامون برای تصحیح خطاهای جزئی بهره می‌برند. سامانه‌های آرایه چندگانه دیسک‌های مستقل از تکنیک‌های مختلف تشخیص و تصحیح خطاها در زمانی که هارد درایو کاملاً قفل کرده‌است، استفاده می‌کنند. حافظه دسترسی تصادفی پویاها نیز از کدهای تصحیح خطا استفاده می‌کنند، تصحیح خطا در این موارد معمولاً از کدهای همینگ استفاده می‌کنند. سامانه‌های کمی هم از شستشوی حافظه استفاده می‌کنند.

تشخیص خطا در شبکه های کامپیوتری

خطا در شبکه‌های کامپیوتری به شرایطی گفته می‌شود که اطلاعات دریافتی با اطلاعات ارسالی مطابقت نداشته باشد. نویزها، مزاحمت زیادی برای سیگنال‌های دیجیتالی در طول زمان انتقال ایجاد می‌کنند یا به عبارتی خطاهایی را برای بیت‌های باینری در حال انتقال به وجود می‌آورند. در این صورت ممکن است یک بیت 0 به بیت 1 تغییر کند و یا بالعکس.

تشخیص خطا در کدها (اجرا شده در لایه Data link یا Transport layer مربوط به مدل OSI)

هر پیغامی که منتقل می‌شود ممکن است در اثر نویز، به هم ریخته شود. جهت جلوگیری از این امر، می‌توان از کدهای تشخیص خطا استفاده نمود. کدهای تشخیص خطا اطلاعات اضافه‌ای هستند که به این پیغام‌های دیجیتالی افزوده می‌شوند تا در صورت بروز خطا در طول زمان انتقال آن را تشخیص دهند.

روش اصلی جهت تشخیص خطا استفاده از بیت‌های افزوده شده است، جایی که بیت‌های مازاد به منظور تسهیل در شناسایی خطا اضافه می‌شوند.

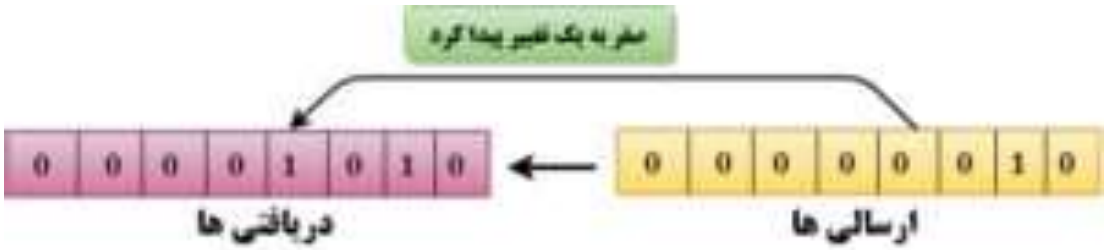
انواع خطاهای شبکه های کامپیوتری

این خطاها را می‌توان به دو دسته تقسیم کرد:

- Single-Bit Error (خطاهای تک بیتی)
- Burst Error (خطاهای متوالی)

Single-bit Error یا خطاهای تک بیتی

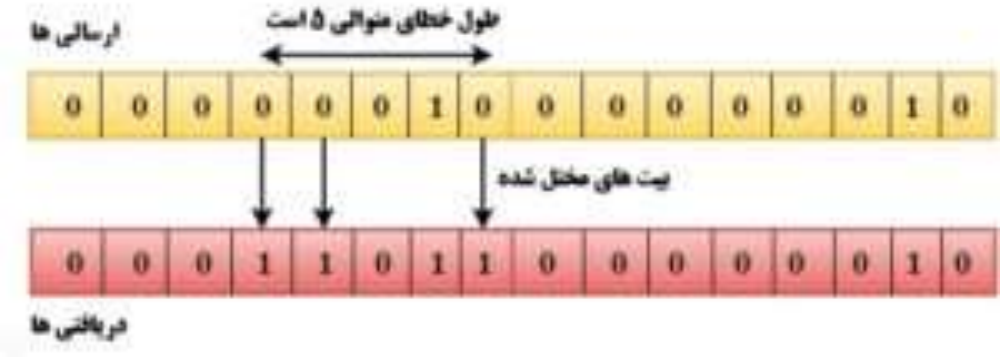
فقط یک بیت از واحدهای داده از یک به صفر یا از 0 به 1 تغییر می‌کند.



Single-bit error عمدتا در انتقال داده‌های موازی (parallel) رخ می‌دهد. به عنوان مثال، اگر از هشت سیم برای ارسال هشت بیت از یک بایت استفاده شود و یکی از سیم‌ها خراب شود، خطای تک بیتی برای هر بایت رخ می‌دهد.

Burst Error یا خطاهای متوالی

خطایی است که در آن چندین بیت از صفر به یک یا از 1 به 0 تغییر می‌کند. خطای Burst یا خطای متوالی، از اولین بیت مختل شده تا آخرین آنها در نظر گرفته می‌شود.



مدت زمان خطای burst نسبت به خطای single bit بیشتر است و عمدتا در انتقال داده‌های Serial رخ می‌دهد. تعداد بیت‌هایی که تحت تاثیر قرار می‌گیرند، بستگی به مدت زمان نویز و میزان داده دارد.

تکنیک‌های تشخیص خطا در شبکه های کامپیوتری

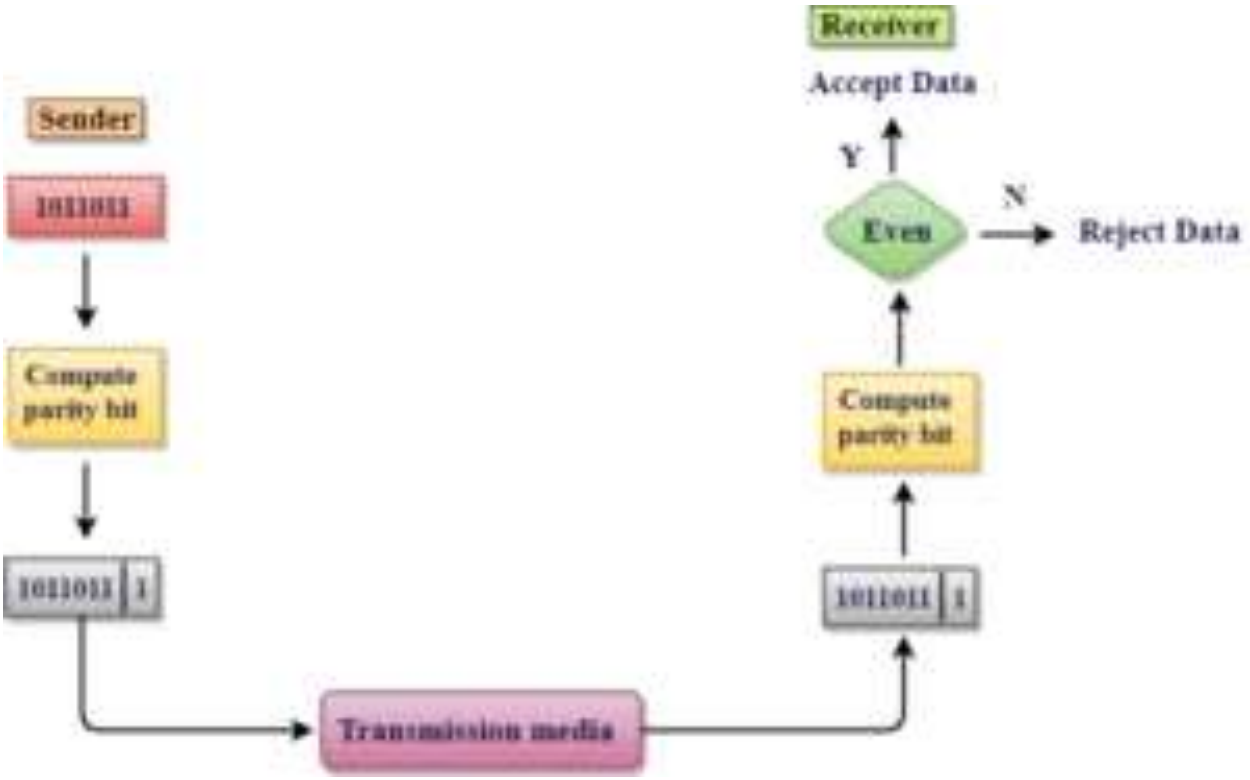
برخی از تکنیک‌های مهم جهت شناسایی خطا عبارتند از:

- Simple Parity check
- Two-dimensional Parity check
- Checksum
- Cyclic redundancy check

1. تکنیکSimple Parity

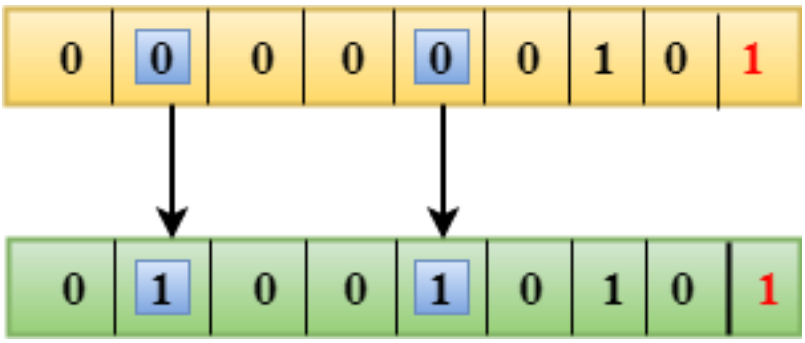
- Parity checking از بیت‌های parity جهت بررسی دقیق انتقال داده‌ها استفاده می‌کند. Parity bit به تمام واحدهای داده (به طور معمول هفت یا هشت بیت) اضافه می‌شود تا داده‌ها منتقل شوند. Parity bit مربوط به هر واحد داده، طوری تنظیم می‌شود که تمامی بایت‌ها یا اعداد فرد داشته باشند و یا مجموعه‌ای از بیت‌های زوج.
- یک مکانیزم ساده و ارزان جهت شناسایی خطاها به شمار می‌رود.

- در این تکنیک، یک بیت مازاد به عنوان parity bit وجود دارد که به انتهای هر واحد داده اضافه می‌شود تا تعداد یک‌ها، زوج شود. بنابراین، تعداد کل بیت‌های منتقل شده 9 خواهد بود.
- در صورتی که تعداد بیت‌های 1 فرد باشد، parity bit 1به آن اضافه می‌شود و در صورتی که تعداد بیت‌های 1 زوج باشد، parity bit 0به انتهای واحد داده اضافه می‌شود.
- در انتهای فرایند دریافت parity bit از داده‌های دریافت شده محاسبه می‌شود و باparity bit های دریافت شده، مقایسه می‌گردد.
- این تکنیک کل 1های فرد را ایجاد می‌کند بنابراین به آن even-parity checking نیز گفته می‌شود.



معایب Simple Parity check–

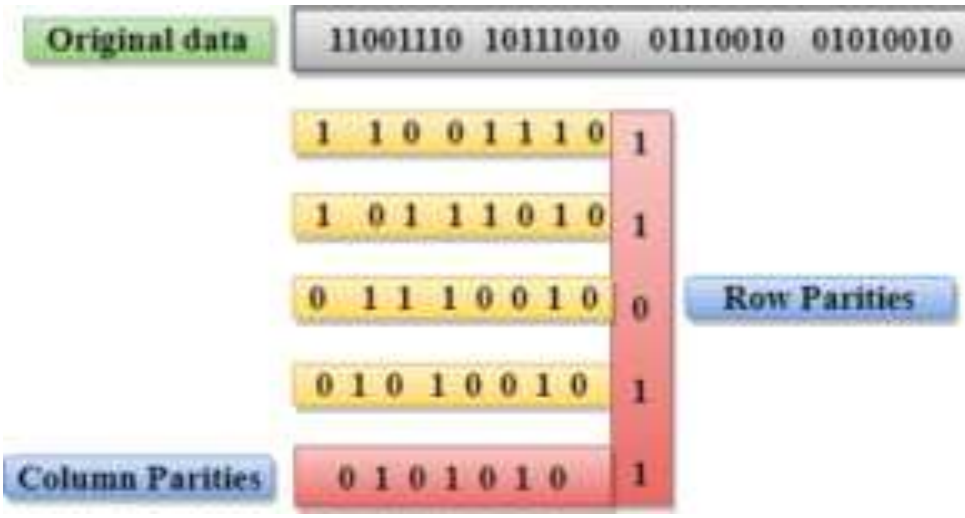
- تنها قادر به شناسایی خطاهای تک بیتی (single-bit) است که بسیار نادر هستند.
- در صورتی که دو بیت با هم عوض شوند، امکان تشخیص خطا را ندارد.



2. تکنیک Two-Dimensional Parity

Two-Dimensional Parityقادر به شناسایی یک یا چند خطای بیت می‌باشد. در صورتی که خطای یک یا چند بیتی رخ دهد، دریافت کننده پیام را با parity bit تغییر یافته دریافت می‌کند. این امر نشان دهنده این است که خطایی رخ داده و شناسایی شده است .

- عملکرد را می‌توان با استفاده از بررسی‌های Two-Dimensional Parity که داده‌ها را درون جدول سازماندهی می‌کند، بهبود بخشید.
- بیت‌های Parity check برای هر ردیف محاسبه می‌شوند که معادل single-parity check است.
- در بررسیTwo-Dimensional Parity ، یک بلوک از بیت‌ها به دو ردیف تقسیم می‌شود و ردیف مازاد بیت‌ها به کل بلوک‌ها اضافه می‌شود.
- در پایان فرایند دریافت، parity bitها با parity bit های دریافت شده از داده‌ها مقایسه می‌شوند.



–معایب بررسیTwo-Dimensional Parity

- چنانچه دو بیت در یک واحد داده مختل شوند و دقیقا دو بیت دیگر با همان موقعیت مکانی در یک واحد داده دیگر نیز مختل گردد، در آن صورت 2 D Parity checkerقادر به شناسایی خطا نخواهد بود.
- در برخی موارد نمی‌توان از این تکنیک جهت شناسایی خطاهای 4بیتی یا بیشتر استفاده کرد.

3. تکنیک Checksum

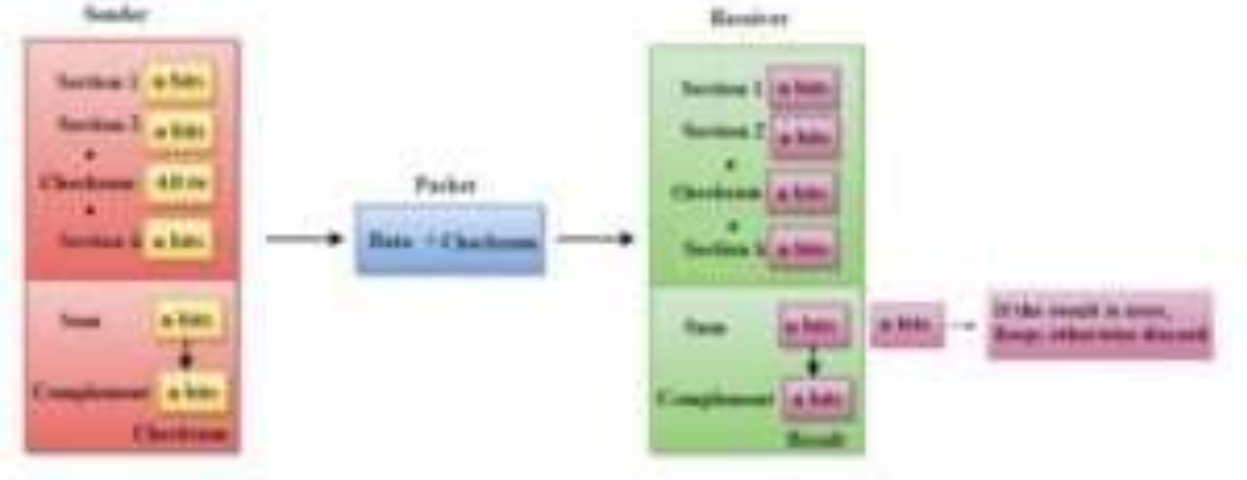
Checksumروشی جهت شناسایی خطاست که بر پایه مفهوم افزونگی طراحی شده است که به دو دسته تقسیم می‌شود:

- **Checksum Generator**

Checksum در قسمت ارسال کننده ایجاد می‌شود. Checksum generator، داده‌ها را به قسمت‌هایی برابر n bit تقسیم می‌کند. سپس همه این قسمت‌ها با استفاده از one's complement arithmetic (مکمل یک) به هم اضافه می‌شوند.

مجموع داده‌ای که به داده‌ی اصلی اضافه می‌شود، به عنوان checksum field معروف است. داده‌های توسعه یافته از طریق شبکه منتقل می‌شوند.

فرض کنید L مجموع کل قسمت‌های داده‌ها می‌باشد، در آن صورت checksum معادل ?L خواهد بود.



فرستنده، مراحل زیر را انجام می‌دهد:

-
  - واحد بلوک به بخش‌های k تقسیم شده و هر یک دارای n bit می‌باشد.

- تمامی قسمت‌های K جهت به دست آوردن مجموع با استفاده از مکمل یک، به هم اضافه می‌شوند.
- حاصل جمع، کامل شده و تحت عنوان checksum field شناخته می‌شود.
- داده‌ی اصلی و checksum field از طریق شبکه، ارسال می‌شوند.

- **Checksum Checker**

Checksumدر قسمت دریافت کننده، تایید می‌شود. دریافت کننده، داده های دریافتی را به قسمت‌هایی برابر با n bit تقسیم می‌کند و تمام این قسمت‌ها به هم اضافه می‌شوند و سپس این حاصل جمع، کامل می‌شود. در صورتی که متمم جمع صفر باشد، داده پذیرفته می‌شود و در غیر این صورت، داده پذیرفته نشده و رد می‌شود.

دریافت کننده، مراحل زیر را انجام می‌دهد:

-
- واحد بلوک به بخش‌های k تقسیم شده و هر یک دارای n bit می‌باشد.
- تمامی قسمت‌های K جهت به دست آوردن مجموع با استفاده از مکمل یک، به هم اضافه می‌شوند.
- حاصل جمع کامل می‌شود.
- در صورتی که حاصل جمع صفر باشد، داده پذیرفته می‌شود و در غیر این صورت، داده‌ها مورد قبول واقع نمی‌شوند.

4. تکنیک Cyclic Redundancy یاCRC

CRCتکنیکی است که جهت تعیین خطا استفاده می‌شود جهت شناسایی سالم بودن داده‌های دریافتی مورد استفاده قرار می‌گیرد .این رویکرد، کار خود را با تقسیم باینری بیت‌های داده‌های ارسالی انجام می‌دهد.

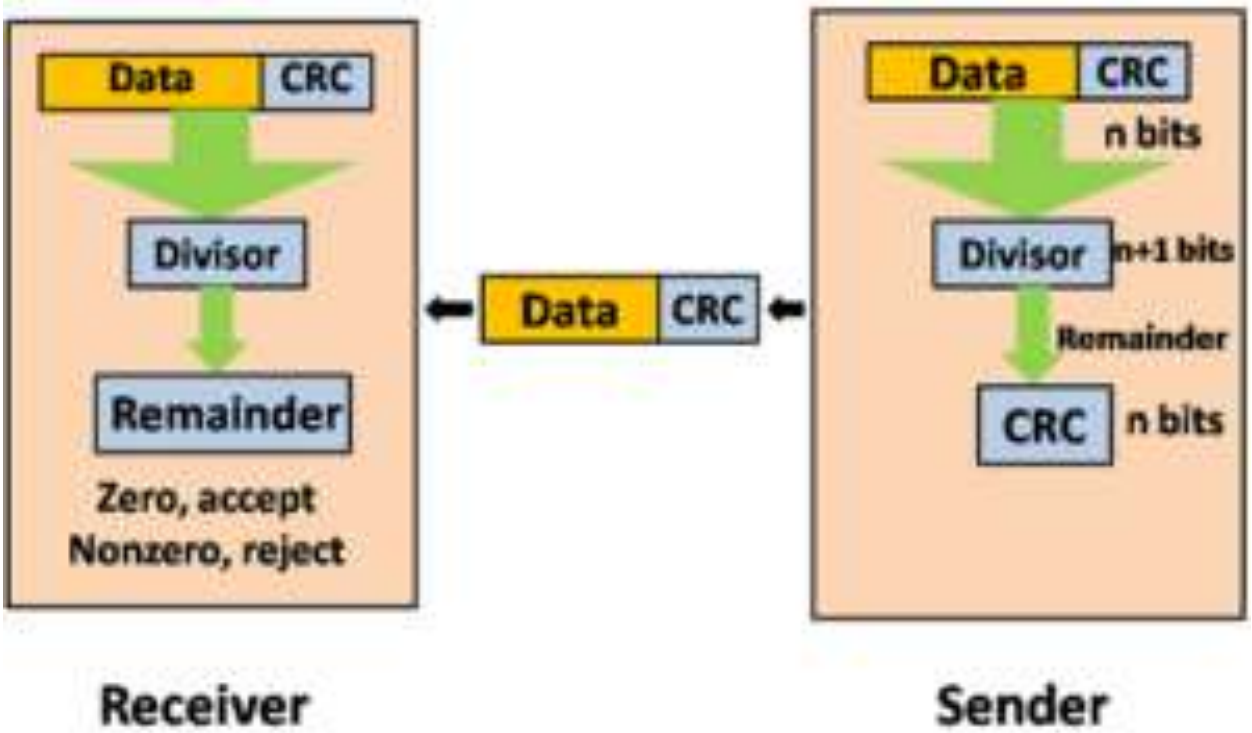
فرستنده عملیات تقسیم را بر روی بیت‌های ارسالی انجام داده و باقی‌مانده را محاسبه نموده و قبل از این که بیت‌های واقعی را بفرستد، باقی‌مانده را به انتهای بیت‌های واقعی می‌افزاید .فرستنده بیت‌های داده‌های واقعی را به همراه باقی‌مانده به صورت رمز ارسال می‌کند. در ضمن در صورتی که خطایی رخ داده باشد، ارسال مجدد انجام می‌شود.

از طرفی دیگر، دریافت کننده پیام نیز عملیات تقسیم را با استفاده از همان مقسوم روی کلمه رمز اجرا می‌نماید. در صورتی که عدد باقی‌مانده با بیت‌های صفر برابر باشد، داده‌ها مورد پذیرش واقع شده و در غیر این صورت داده‌ها از بین می‌روند.

مراحل تشخیص خطا در تکنیک:CRC

- در تکنیکCRC ، یک رشته از تعدادn های صفر به واحد داده‌ها پیوست می‌شود و این تعدادn ، کمتر از تعداد بیت‌های از پیش تعیین شده است، به عبارتی n+1بیت
- داده‌هایی که اخیرا گسترش پیدا کرده‌اند با استفاده از یک فرایند و توسط یک تقسیم کننده، تقسیم می‌شوند که تحت عنوان binary division معروف است. باقی‌مانده‌ی تولید شده از این تقسیم نیز CRC remainder نامیده می‌شود.
- CRC remainderتوسط صفرهای پیوست شده به انتهای داده‌های اصلی، جایگزین می‌شوند. این واحد تازه تولید شده به دریافت کننده ارسال می‌شود.
- دریافت کننده، داده‌های دنبال شده توسط CRC remainder را دریافت می‌کند. در ضمن دریافت کننده با کل این واحد مانند یک single unit رفتار می‌کند و توسط همان تقسیم کننده‌ای که جهت یافتن CRC remainder مورد استفاده قرار گرفته، تقسیم می‌شود.

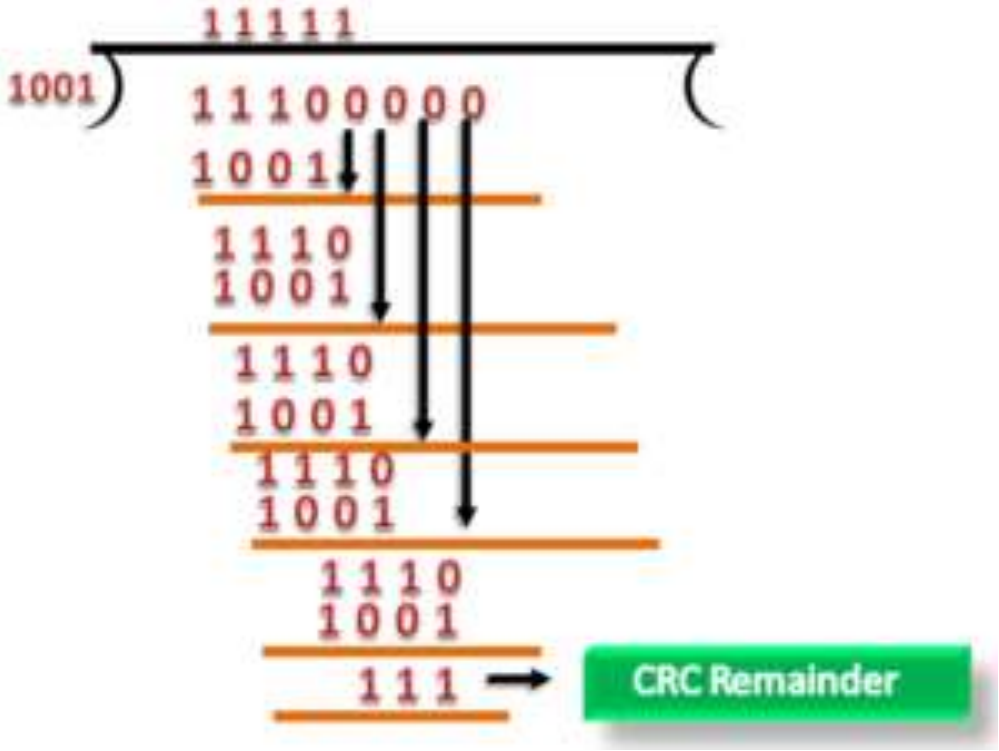
چنانچه نتیجه این تقسیم صفر باشد، به این معناست که هیچ‌گونه خطایی وجود ندارد و داده‌ها پذیرفته می‌شوند. در صورتی که نتیجه‌ی این تقسیم صفر نباشد به این معنی است که خطایی در داده‌ها وجود دارد. بنابراین داده‌ها پذیرفته نمی‌شوند.



حال این مفاهیم را از طریق یک مثال بیان می‌کنیم.

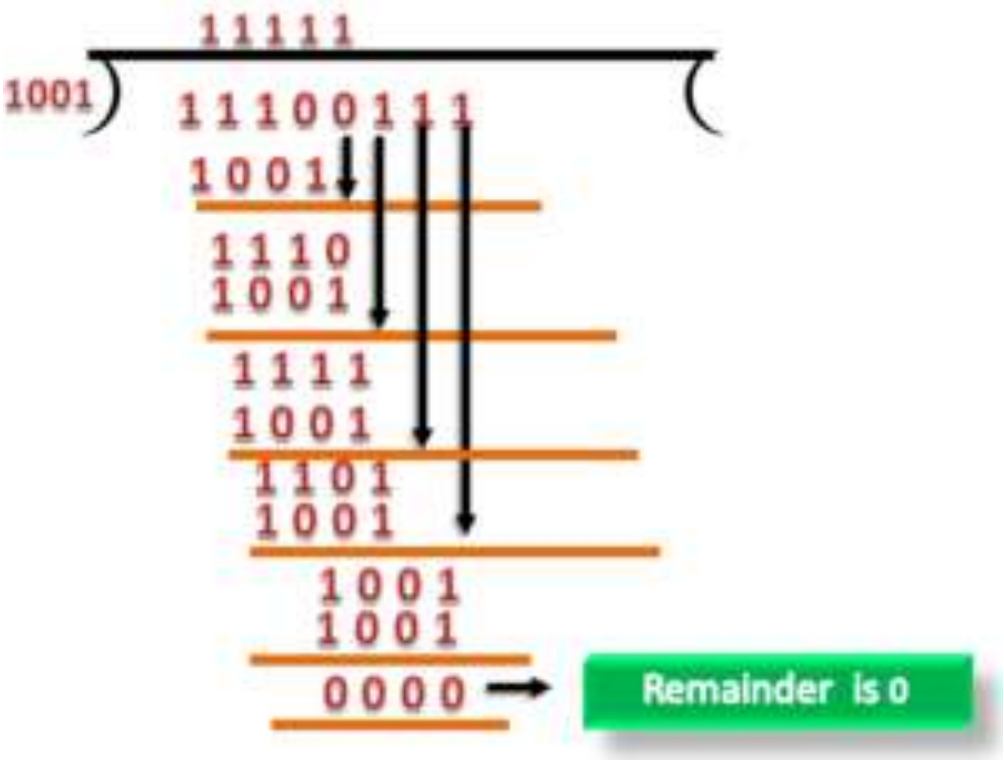
فرض کنید داده اصلی 11100 است و تقسیم کننده 1001

- **CRC Generator**
-
  - یک CRC generator از تقسیم modulo-2 استفاده می‌کند. در ابتدا سه صفر به انتهای داده پیوست می‌شود تا طول تقسیم کننده 4 شود و همان‌طور که می‌دانیم طول رشته‌های صفرهایی که پیوست می‌شوند همیشه یکی کمتر از طول تقسیم کننده می‌باشد.
  - حال عدد ما می‌شود 11100000 و عدد به دست آمده تقسیم بر تقسیم کننده 1001 می‌شود.
  - باقی‌مانده به دست آمده از این تقسیم باینری تحت عنوان CRC remainder شناخته می‌شود. مقدار عددی تولید شده ازCRC remainder ، عدد 111 است.
  - CRC remainderتوسط رشته‌ای از اعداد، صفر را به انتهای واحد داده پیوست می‌کند و رشته نهایی 11100111 خواهد بود که از طریق شبکه ارسال می‌شود.



- **CRC Checkerدر تشخیص خطا در شبکه های کامپیوتری**
-
  - عملکرد CRC checker مشابه عملکرد CRC generator است.

- زمانی که رشته‌ی اعداد 11100111 دریافت می‌شود، سپس CRC checker تقسیم modulo-2را انجام می‌دهد.
- رشته‌ی اعداد به همان تقسیم کننده، یعنی 1001 تقسیم می‌شود.
- در این حالت، CRC checkerباقی‌مانده‌ی صفر را ایجاد می‌کند. بنابراین، داده پذیرفته می‌شود.



اصلاح خطا در شبکه های کامپیوتری

مرحله‌ی بعد از تشخیص خطا در شبکه‌های کامپیوتری، اصلاح آنها می‌باشد. به هنگام ارسال داده از فرستنده به گیرنده، کدهای اصلاح خطا جهت شناسایی و تصحیح خطاها مورد استفاده قرار می‌گیرند. اصلاح خطا در شبکه های کامپیوتری را می‌توان به دو روش زیر انجام داد:

- **Backward error correctionیا اصلاح خطای رو به عقب**

گیرنده پس از شناسایی و کشف خطا، از فرستنده درخواست مجدد کل داده‌ها را دارد.

- **Forward error correctionیا اصلاح خطا رو به جلو**

در این حالت، پس از شناسایی خطا فرستنده از کد اصلاح خطا استفاده نموده که به کمک آن خطاها به صورت اتوماتیک، تصحیح می‌شوند.

یک بیت اضافه می‌تواند خطا را تشخیص دهد اما نمی‌تواند آن را اصلاح کند.

به منظور اصلاح خطا، محل دقیق آن باید مشخص باشد. به عنوان مثال، در صورتی که قصد داشته باشیم خطای تک بیتی را محاسبه نماییم، کد اصلاح خطا مشخص می‌کند که کدام یک از هفت بیت دارای خطا است. جهت دستیابی به این هدف می‌بایست چندین بیت اضافه، بیافزاییم.

فرض کنید(r) ، تعداد بیت‌های اضافه و (d) تعداد کل بیت‌های داده می‌باشد. تعداد بیت‌های مازاد r را می‌توان با استفاده از این فرمول محاسبه کرد:

$$2^r >= d+r+1$$

مقدار r را می‌توان با استفاده از فرمول بالا محاسبه کرد. به عنوان مثال، در صورتی که ارزش d چهار باشد، کمترین مقدار ممکن که می‌توان در این فرمول قرار داد، سه است.

R.W Hammingیک تکنیک جهت تعیین موقعیت بیتی (bit) که خطا دارد، تحت عنوان کد Hamming، تعریف کرده است. در ضمن می‌توان آن را در هر طول واحد داده اجرا کرد و از رابطه‌ی بین واحدهای داده و واحدهای اضافه استفاده کرد.

کدHamming

- **Parity bits:**بیتی است که به داده اصلی بیت‌های باینری، پیوست می‌شود تا تعداد کل 1ها فرد یا زوج باشد.
- **Even parity:**به منظور بررسیeven parity ، در صورتی که تعداد کل 1ها زوج باشد، مقدار عددی parity bit برابر صفر است. چنانچه تعداد کل 1ها فرد باشد، مقدار عددی parity bit برابر یک خواهد بود.
- **Odd Parity:**به منظور بررسیodd parity ، در صورتی که تعداد کل 1ها زوج باشد، مقدار عددی parity bit برابر یک است. چنانچه تعداد کل 1ها فرد باشد، مقدار عددی parity bit برابر صفر خواهد بود.

الگوریتم کد همینگ در اصلاح خطا در شبکه های کامپیوتری

- اطلاعات بیت‌های "d" به بیت‌های افزونه "r" اضافه می‌شوند تا d+r را تشکیل دهند.
- مکان تمامی ارقامd+r ، عدد اعشاری اختصاص داده می‌شود.
- بیت‌های "r" در موقعیت مکانی 2،1،... 2^{k-1}قرار می‌گیرند.
- در پایان فرایند دریافت، parity bitها محاسبه می‌شوند. مقدار عددی اعشاریparity bit ها، خطا را تعیین می‌کنند.

رابطه‌ی بین موقعیت مکانی خطا (Error Position) و اعداد باینری(Binary Number)

Error Position	Binary Number
0	000
1	001
2	010
3	011
4	100
5	101
6	110
7	111

حال مفهوم کد همینگ را در قالب یک مثال، توضیح می‌دهیم.

فرض کنید داده‌ی اصلی که قرار است ارسال شود، 1010 است.

تعداد کل بیت‌های داده‌ی d =4l

تعداد بیت‌های مازاد r: l d+r+12 =< d+r+112lr >= d+r+2r >= d+r+12r >= d+r+12 2r >= d+r+1 r

$$r >= 4+r+12$$

بنابراین، مقدار عددی r که بتوان در فرمول بالا گذاشت، 3 می‌باشد.

تعداد کل بیت‌ها = d+r = 4+3 = 7

تعیین موقعیت بیت‌های مازاد

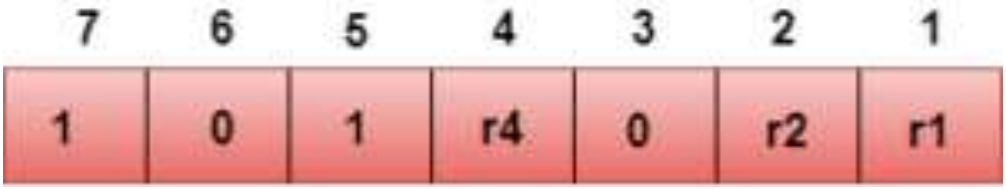
تعداد بیت‌های مازاد، 3 است. این سه بیت به صورت r1, r2, r4 نشان داده می‌شوند. موقعیت مکانی بیت‌های مازاد بعد از رسیدن به توان دو محاسبه می‌شود. بنابراین، موقعیت متناظر آنها 1, 2^1 ,2^2در نظر گرفته می‌شود.

موقعیت = r1 l1

موقعیت = r2 l2

موقعیت = r4 l4

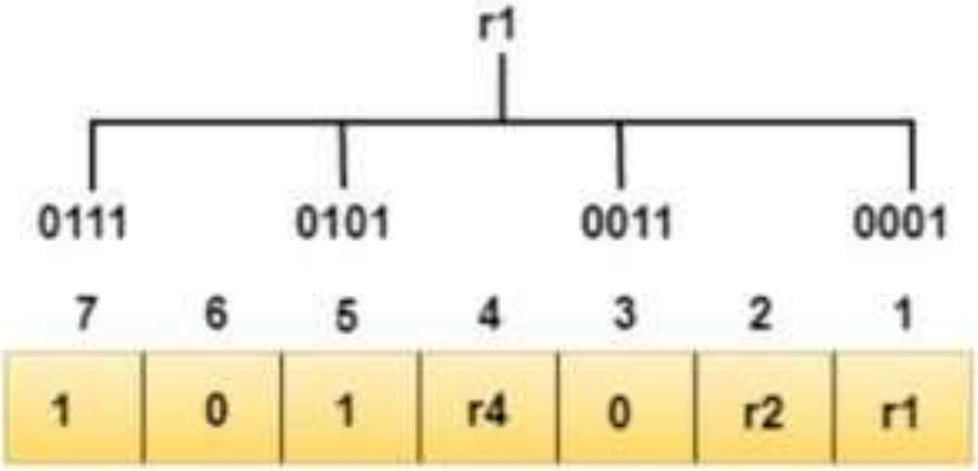
نمایش داده‌ها بر روی parity bit های اضافه شده



مشخص کردنParity bits

–تعیین بیت r1

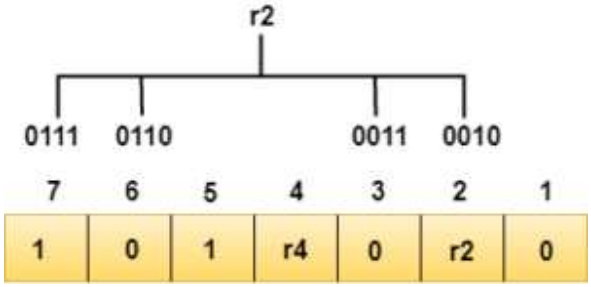
بیت r1 را می‌توان با انجام parity check بر روی بیت‌هایی که موقعیت آنها در نمایش باینری شامل یک در موقعیت اول می‌باشد محاسبه نمود.



همان طور که از شکل بالا مشخص است، bit positionهایی که شامل 1 در موقعیت اول هستند، 1 ، 3 ، 5 و 7 می‌باشند. حال even-parity check را بر روی موقعیت این بیت‌ها انجام می‌دهیم. تعداد کل 1ها در این موقعیت‌های بیت مربوط به r1 ، زوج است بنابراین، مقدار بیت r1 برابر صفر است.

–تعیین بیت r2

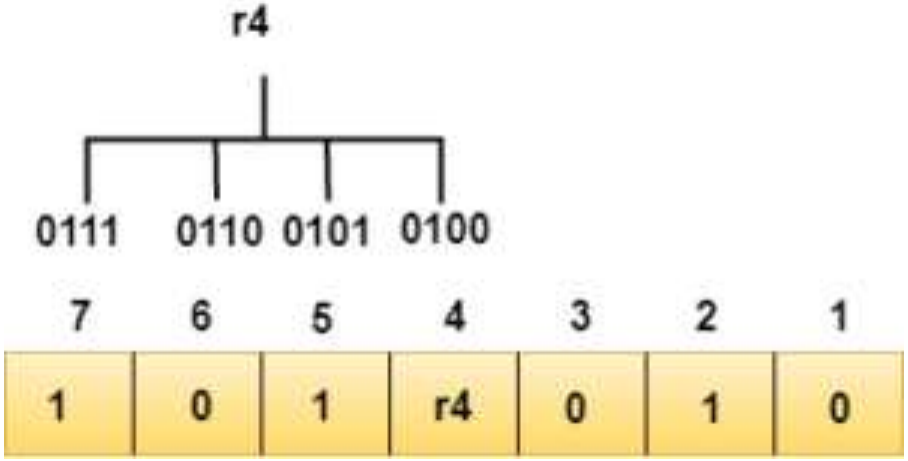
بیت r2 را می‌توان با انجام parity check بر روی بیت‌هایی که موقعیت آنها در نمایش باینری شامل یک در موقعیت دوم می‌باشد محاسبه نمود.



همان طور که از شکل بالا مشخص است، bit positionهایی که شامل 1 در موقعیت دوم هستند، 2 ، 3 ، 6 و 7 می‌باشند. حال even-parity checkرا بر روی موقعیت این بیت‌ها انجام می‌دهیم. تعداد کل 1ها در این موقعیت‌های بیت مربوط به r2 ، فرد است، بنابراین مقدار بیت r2 برابر یک است.

–تعیین بیت r4

بیت r4 را می‌توان با انجام parity check بر روی بیت‌هایی که موقعیت آنها در نمایش باینری شامل یک در موقعیت سوم می‌باشد محاسبه نمود.



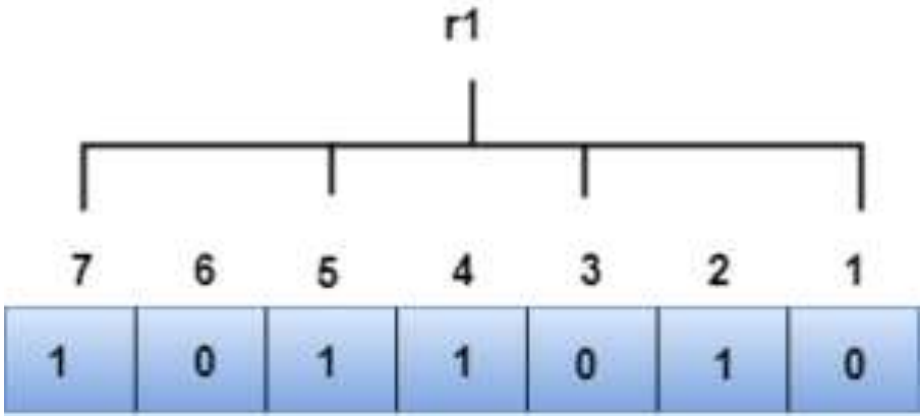
همان طور که از شکل بالا مشخص است، bit positionهایی که شامل 1 در موقعیت سوم هستند، 4 ، 5 ، 6 و 7 می‌باشند. حال even-parity checkرا بر روی موقعیت این بیت‌ها انجام می‌دهیم. تعداد کل 1ها در این موقعیت‌های بیت مربوط به r4 ، زوج است بنابراین مقدار بیت r4 برابر صفر است.

داده‌های انتقال یافته را در این قسمت می‌توانید مشاهده نمایید:

فرض کنید وقتی این داده‌ها به گیرنده رسید بیت چهارم از صفر به یک تغییر کند، در این صورت parity bitها مجددا محاسبه می‌شوند.

بیت R1

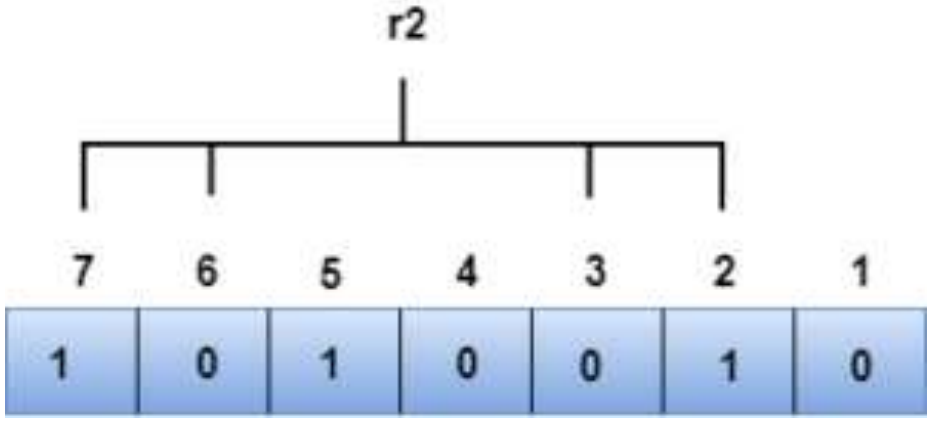
موقعیت بیت‌های r1 در جایگاه 1، 3 ، 5 و 7 می‌باشد.



همان طور که از شکل بالا مشخص است، نمایش باینری R1 عدد 1100 می‌باشد. حال ما even-parity checkرا انجام می‌دهیم، مجموع کل 1های پیوست شده به بیت r1 یک عدد زوج است. بنابراین مقدار r1برابر صفر می‌باشد.

بیتR2

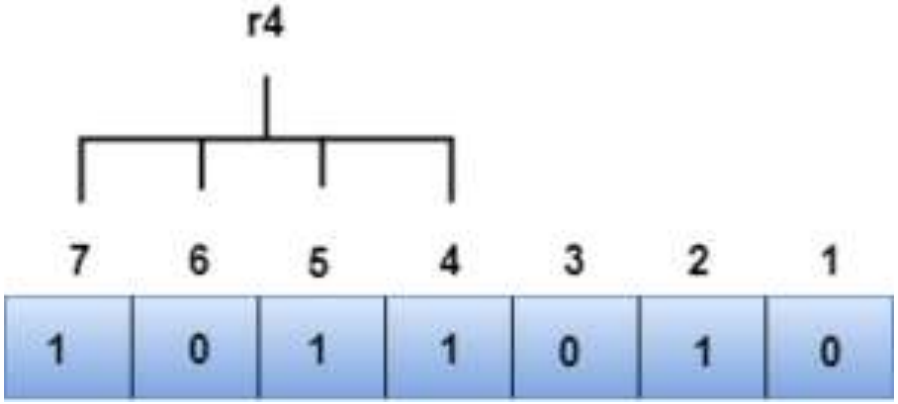
موقعیت بیت‌های r2 در جایگاه 2 ، 3 ، 6 و 7 می‌باشد.



همان طور که از شکل بالا مشخص است، نمایش باینری R2 عدد 1001می‌باشد. حال ما even-parity checkرا انجام می‌دهیم، مجموع کل 1های پیوست شده به بیت r2 یک عدد زوج است. بنابراین مقدار r2برابر صفر می‌باشد.

بیتR4

موقعیت بیت‌های r4 در جایگاه 4 ، 5 ، 6 و 7 می‌باشد.



همان طور که از شکل بالا مشخص است، نمایش باینری R4 عدد 1011 می‌باشد. حال ما even-parity checkرا انجام می‌دهیم، مجموع کل 1های پیوست شده به بیت r4 یک عدد فرد است. بنابراین مقدار r4برابر یک می‌باشد.

بررسی انواع کدهای تشخیص و تصحیح خطا Decoding & Coding

هرگاه یک کانال ارتباطی برای انتقال اطلاعات داشته باشیم در حین انتقال به دلیل وجود نویز اطلاعات دچار تغییر می شوند. باید روشی برای مشخص کردن این تغییرات داشته باشیم و بهتر است به روشی دست یابیم که میتواند این تغییرات ناخواسته یا خطا ها را اصلاح نماید.

1-1 مفاهیم کدینگ (Coding Concepts)

برای آنکه بتوانیم یك کلمه (Word) از داده ها را بگونه ای کد گذاری کنیم که قابلیت تشخیص و تصحیح خطا را داشته باشد، باید تعداد بیت های آن را افزایش دهیم. اگر طول یك Data Word به اندازه D بیت باشد، پس از کد گذاری یك کلمه کد شده (Codeword) به اندازه C بیت خواهد بود. بگونه ای که D<C می‌باشد. پس حالا ما بجای D2 حالت ممکن، C2 حالت ممکن داریم. ولی تمام این حالت ها درست نیستند، و این همان چیزی است که باعث می شود سیستم بتواند وجود خطا را تشخیص دهد. یعنی اگر یك عدد در یکی از این حالات غیرمجاز باشد، سیستم می فهمد که خطایی روی داده است. در بعضی از روش ها، سیستم در یك سری از حالات می تواند خطای بوجود آمده را نیز اصلاح کند. روش ارائه شده باید این قابلیت را داشته باشد که از بین C بیت موجود D بیت اصلی را خارج کند. به این عمل اصطلاحا Decoding می گویند. یکی از مشکلات استفاده از کدینگ این است که سیستم مجبور است تا یك مدت زمانی را صرف عملیات Encoding و Decoding کند که باعث ایجاد سربار (Overhead) در سیستم می شود.

2-1 کد همینگ

در دهه ۱۹۵۰ میلادی ریچارد همینگ که در آزمایشگاههای شرکت بل کار می کرد به معرفی دسته ای از کد های اصلاح کننده خطا پرداخت که بنام خود او کدهای همینگ خوانده می شوند. شاید ساده ترین روش برای آشکار کردن خطای یک بیت در یک بایت، استفاده از بیت توازن است.

3-1 فاصله همینگ (Hamming Distance)

فاصله همینگ بین دو Codeword برابر است با تعداد بیت هایی که آنها با هم متفاوتند. یعنی نشان میدهد که اگر در اثر خطا یك کد بخواهد به یك کد دیگر تبدیل شود، چند بیت از آن باید تغییر کند تا این تبدیل انجام شود بدون آنکه سیستم آن را خطا به حساب آورد .
در تئوری اطلاعات فاصله همینگ بین دو رشته برابر طول تعداد مکانهایی است که سمبولهای متناظر متفاوت هستند. به معنای دیگر، کمترین تعداد جایگزینی هایی است که یک رشته به یک رشته دیگر تغییرپیدا کند، یا تعداد خطاهایی که یک رشته به رشته دیگر تبدیل گردد.

چند مثال برای فاصله همینگ بین چند رشته:
«toned»و«roses» فاصله همینگ سه هست.
۱۰۱۱۱۰۱ و ۱۰۰۱۰۰۱ فاصله همینگ دو هست.
۲۱۷۳۸۹۶ و ۲۲۳۳۷۹۶ فاصله همینگ سه هست.

کدهای 101 و 011 در 2 بیت با یك دیگر متفاوت هستند. در نتیجه فاصله همینگ بین آنها برابر 2 است.

اما کدهای 101 و 100 فقط در یك بیت با هم تفاوت دارند. در نتیجه اگر یك خطا در بیت کم ارزش آنها روی دهد، یکی از آنها را به دیگری تبدیل می کند و سیستم متوجه وجود خطا نخواهد شد. فاصله همینگ به اندازه 2 تضمین می کند که اگر یك خطای تك بیتی اتفاق بیفتد سیستم حتما متوجه بروز خطا خواهد شد.
در شکل روبه رو مکعب باینری را میبیند که در هر گوشه آن یک عدد باینری قرار دارد . در این مکعب هر ضلع یک فاصله همینگ به حساب می آید . برای مثال فاصله بین دو عدد 001 تا 010 دو ضلع است به عبارتی فاصله همینگ آن 2 است .

4-1 فاصله کد (Code Distance)

فاصله کد برابر است با کمترین فاصله همینگ که بین هر دو کد موجود در یك مجموعه کد وجود دارد. یعنی اگر مثلا در یك روش کدینگ فاصله کد برابر 2 باشد به این معنی است که هیچ کدام از کدها با کدهای دیگر فاصله همینگ کمتر از 2 ندارند. برای مثال مجموعه کدهای {001، 010، 100، 111}

همگی باهم فاصله 2 دارند. در نتیجه این کد می تواند هر خطای تك بیتی را تشخیص دهد.
به عنوان مثالی دیگر کدهای {000، 111} دارای فاصله 3 هستند پس می توانند هر خطای تك بیتی یا دو بیتی را تشخیص دهند. اما اگر فرض شود احتمال خطای دو بیتی کم است، این کد را می توان به عنوان روشی که می تواند خطاهای تك بیتی را اصلاح (Correct) کند، نیز استفاده شود.

5-1 محدودیت تشخیص و تصحیح (Detection and Correction)

به عنوان یك تعریف ریاضی می توان گفت : برای آنکه بتوانیم تا حداکثر t بیت خطا را تشخیص دهیم، نیاز به حداقل فاصله کد به اندازه t+1 داریم. ولی برای آنکه بتوانیم تا حداکثر t بیت خطا را تصحیح کنیم، نیاز به حداقل فاصله کد t+12 داریم.

6-1 کدینگ و افزونگی (Coding and Redundancy)

فرض کنید که یك مجموعه کد شامل دو حالت به صورت {000، 111} باشد که برای نشان دادن تنها یك بیت به کار می رود. در واقع عدد 0 به شکل 000 کد شده است و عدد 1 به شکل 111. این سیستم کد دهی معادل سیستم های TMR می باشد. در واقع کدینگ همیشه همراه با افزونگی (Redundancy) می‌باشد که در نتیجه می توان از تکنیکهای بکار رفته شده برای افزونگی در کدینگ نیز استفاده کرد. مثلا Duplex یکی از راه های افزونگی است که در این روش Codeword دو بار عینا تکرار می شود. برای مثال برای یك تك بیت دو حالت وجود دارد که 00 و 11 است که از دو بار تکرار 0 و 1 به دست آمده اند.

7-1 جداپذیری کد (Code Separability)

داده های کد شده می توانند دو حالت داشته باشند :

جدا پذیر (Separable)

کدی را جداپذیر می گوییم که بیت های مربوط به داده اصلی با بیت های اضافه شده برای کد از هم جدا باشند. در این حالت استخراج اطلاعات از کد بسیار ساده تر است. چون تنها کافیست که بیت های مربوط به کد را کنار بگذاریم.

جدا ناپذیر (Non-Separable)

در کدهای جداناپذیر داده های اصلی با کدهای اضافی با هم ترکیب شده اند و جدا سازی آنها از یك دیگر نیاز به انجام پردازش های اضافی دارد.

1-2 روشهای کدینگ (Coding methods)

1-1-2 کد Parity (Parity Coding)

پریتی (Parity) ساده ترین روش کد گذاری جدا پذیر است. در این روش اطلاعات کد شده شامل N بیت داده اصلی به همراه یك بیت اضافه که Parity را نگه می دارد، می‌باشد. دو نوع Parity وجود دارد:

Even (زوج)

در روش زوج بیت Parity به گونه ای تنظیم می شود که تعداد یك ها در کل بیت ها (داده اصلی و Parity) زوج باشد.

Odd (فرد)

روش فرد بر عکس عمل می کند. یعنی در روش فرد بیت Parity به گونه ای تنظیم می شود که تعداد یك ها در کل بیت ها (داده اصلی و Parity) فرد باشد.
تعداد کل بیت ها در نهایت برابر (N+1) است. در این حالت عملا به میزان N/1 بیت جدید به داده اضافه شده است. کد Parity دارای فاصله همینگ 2 می‌باشد که در نتیجه می تواند هر خطای تك بیتی را تشخیص دهد ولی نمی تواند هیچ نوع تصحیحی انجام دهد. کد Parity نمی تواند یك خطای دو بیتی را تشخیص دهد، ولی خطاهای سه بیتی را می تواند تشخیص دهد. در کل کد پریتی قابلیت تشخیص خطا در تعداد فرد را دارد .

Parity فرد بهتر است یا زوج؟

اینکه کدام یك از دو حالت Parity موثرتر هستند کاملا بستگی به شرایط دارد. یکی از خطاهای رایج به نام Burst Error یا Error All-Bits وجود دارد . در این نوع خطا همه بیت ها یا 1 می شوند و یا 0 می شوند . در صورتی که از Parity زوج استفاده شود آنگاه خطای همه 0 (All-0'S) قابل تشخیص نیست. ولی با انتخاب Parity فرد این خطا تشخیص داده می شود . پس اگر احتمال خطای همه 0 بیشتر است بهتر است که از Parity فرد استفاده شود. اگر احتمال خطای همه 1 بیشتر است، آنگاه دو حالت وجود دارد اگر تعداد کل بیت ها (همراه با Parity، N+1) زوج باشد باید از Parity فرد و اگر تعداد کل بیت ها فرد باشد از Parity زوج استفاده کرد .

می توانیم بجای آنکه به کل بیت ها یك Parity اختصاص دهیم به هر گروه از آنها، مثلا هر یک بایت، یك Parity اختصاص دهیم. در این حالت بدیهی است که میزان Overhead از N/1 به M/N افزایش خواهد یافت. (M تعداد گروه یا بایت ها است) در این حالت حد اکثر M خطا قابل تشخیص است، البته به شرطی که خطا ها در بایت های مختلف باشند. اگر هر دو نوع خطای همه 0 و همه 1 ممکن است اتفاق بیفتد می توانید از پریتی Parity برای یك بایت و از Parity فرد برای بایت بعدی استفاده کنید.

2-1-2 کد همینگ

در اصل کد همینگ یک نوع کدگذاری از خانواده ی کدگذاری پریتی است . در روش همینگ از سه بیت توازن برای آشکارسازی و اصلاح خطا استفاده میشود. همانطور که در شکل مشخص است چهار بیت

d1 الی d4 به عنوان داده ورودی در نظر گرفته میشوند. سپس با ترتیب نشان داده شده بیتهای توازن p1 تا p3 از XOR کردن بیت ها محاسبه می شوند و در نهایت داده هفت بیتی بدست آمده ارسال می گردد.

نحوه محاسبه بیتهای توازن در کد همینگ

نمایش گرافیکی از 4 بیت اطلاعات و 3 بیت پریتی که نشان می دهد کدام بیت داده در کدام بیت پریتی اثر گذار است.

در مقصد بیت توازن با بیتهای گروه خود XOR میشود مثلا بیتهای p1 و d1 و d2 و d4 با هم XOR می شوند و نتیجه به عنوان بیت اول نشانه s1 در نظر گرفته میشود به همین ترتیب بیتهای دوم و سوم نشانه هم بدست می آیند. هرگاه هر سه بیت نشانه صفر باشد داده درست منتقل شده است. اما در صورت یک بودن هر یک از بیت های خطا رخ داده است. اگر سه بیت نشانه را از کوچک به بزرگ در کنار هم قرار دهیم یک عدد سه بیتی بدست می آید که مقدار آن نشان دهنده محل وقوع خطاست . با عوض کردن بیت مورد نظر داده اولیه بدست می آید. باید توجه داشت که این روش همینگ امکان اصلاح یک خطا را دارد و در صورت بروز دو خطا فقط امکان آشکار سازی وجود دارد

خطا در بیت ششم رخ داده است

برای آنکه بدانیم به چند بیت برای Parity نیاز داریم ، باید طبق رابطه زیر عمل کنیم : اگر تعداد بیت های داده برابر D باشد و تعداد بیت های Parity برابر R باشد. در آن صورت جمعا" D+R بیت داریم که هر کدام از آنها می تواند دچار خطا شود یعنی با فرض اینکه خطا های ما تك بیتی هستند، D+R حالت مختلف خطا داریم. علاوه بر حالت های خطا یك حالت درست هم داریم که در آن هیچ بیتی دچار اشکال نشده است. پس جمعا D+R+1 حالت ممکن وجود دارد که باید توسط Parity نمایش داده شود. پس با توجه به اینکه R بیت Parity وجود دارد می توانیم R2 حالت مختلف داشته باشیم که شامل حالت های خطا و درست می شود. پس اگر داشته باشیم :

$$R+D+1 =< R2$$

آنگاه می توانیم مطمئن باشیم که تعداد بیت های Parity کافی است.

3-1-2 جمع کنترلی (Checksum)

این روش در اصل برای سیستمهای انتقال اطلاعات استفاده می شود. ایده اصلی آن این است که بایت های یك بلوك از داده ها با یك دیگر جمع شوند و حاصل جمع نیز ارسال شود. گیرنده نیز داده ها را جمع می کند و اگر با حاصل جمع دریافتی یکی نباشد، می فهمد که خطا روی داده است. گونه های مختلفی برای Checksum وجود دارد که در اینجا آنها را بررسی می کنیم: (فرض کنیم که هر کلمه از داده ها دارای طول D باشد).

2-1-3-1 Single-Precision (تک دقتی):
در این روش جمع به پیمانه 2D (Modulo) انجام می شود. یعنی حاصل جمع به D2 تقسیم می شود و باقیمانده آن فقط در نظر گرفته می شود. یا به عبارتی تنها D رقم سمت راست حاصل جمع در نظر گرفته می شود.

2-1-3-2 Double-Precision (دقت مضاعف):
کاملا شبیه Single است ولی بجای D2 از D22 استفاده می شود که در نتیجه این روش خطاهای بیشتری را می تواند کشف کند.

2-1-3-3 Residue Checksum (باقیمانده):
در این روش بیت های اضافی بعد از D اُمین بیت که در روش Single دور ریخته می شد، مجددا با خود داده اصلی جمع می شود که در نتیجه قابلیت اطمینان سیستم بالاتر می رود. زیرا وجود خطا در آن بیت های اضافی نیز تاثیر گذار هستند.

2-1-3-4 Honeywell Checksum :
در این روش هر دو کلمه را به هم می چسبانند و سپس کل این مجموعه های دوتایی را با هم جمع میکنند و نتیجه را به پیمانه D22 در نظر می گیرند. حسن این روش این است که اگر یك خطا همواره روی یکی از بیت های هر کلمه (مثلا بیت سوم) اتفاق بیفتد، در گونه های قبلی ممکن بود تشخیص داده نشود، ولی در این روش جلوی این نوع خطا ها نیز گرفته می شود.

نکته : روشهای Checksum فقط می توانند وجود خطا را تشخیص دهند ولی نمی توانند آن را تصحیح کنند. به همین خاطر اگر خطایی روی دهد، کل بلوك باید مجددا ارسال شود.

2-1-4 کد برگر (Berger Code)
کد بِرگر یك روش جداپذیر (Separable) است. این روش به این شکل عمل می کند که ابتدا تعداد یك های درون داده را می شمارد، سپس از عدد به دست آمده مکمل می گیرد و سپس این عدد به دست آمده را در کنار عدد اصلی قرار میدهد.
برای مثال فرض کنید عدد 11101 را داریم. درون این عدد چهار 1 وجود دارد که فرم باینری آن 100 می شود و مکمل آن 011 است. حالا اگر این عدد را در کنار عدد اصلی قرار دهیم، داریم 11101011 . این روش می تواند هر نوع خطای Unidirectional را تشخیص دهد، چه خطا از 0 به 1 باشد یا برعکس آن. اما اگر هم زمان بعضی 0 ها به 1 تبدیل شوند، و همان تعداد 1 نیز به 0 تبدیل شوند، نمی تواند خطا را تشخیص دهد.

2-1-5 کد افزونگی چرخشی CRC

یک کد افزونگی چرخشی (به انگلیسی: code Cyclic redundancy) (سی‌آرسی) تابع درهم‌سازی غیرایمنی است که جهت تشخیص تغییرات تصادفی رو داده‌های خام طراحی شده‌است. این تابع عموما در شبکه‌های مخابراتی دیجیتال و وسایل ذخیره‌سازی داده‌ها از جمله دیسک سخت مورد استفاده قرار می‌گیرد. یک دستگاه دارای قابلیت سی‌آرسی، یک توالی کوتاه و با طول ثابت را، به نام کد سی‌آرسی (یا فقط سی‌آرسی)، برای هر بلاک از داده‌ها محاسبه نموده و آن را همراه با داده‌ها ذخیره یا ارسال می‌کند. زمانی که یک بلاک دریافت یا خوانده می‌شود دستگاه محاسبه را تکرار می‌کند؛ در صورت مغایرت با کد محاسبه شده قبلی مشخص می‌شود که این بلاک دارای خطای داده است و در این حالت دستگاه ممکن است عملی را جهت اصلاح خطا از جمله خواندن یا درخواست ارسال مجدد بلاک انجام دهد. اصطلاح سی‌آرسی می‌تواند به کد اعتبارسنج یا تابع تولید کد اطلاق شود. سی‌آرسی‌ها به جهت پیاده‌سازی ساده در سخت‌افزار دودویی، سادگی تحلیل ریاضی آن‌ها و عملکرد خوب در تشخیص خطاهای معمول حاصل از اختلال در کانال‌های انتقال دارای محبوبیت زیادی هستند. سی‌آرسی توسط W. Wesley Peterson اختراع و در مقاله ۱۹۶۱ وی منتشر شد . سی‌آرسی 32 بیتی پیشنهادی موسسه مهندسین الکتریک و الکترونیک (IEEE)، که در اترنت و سایر جاها استفاده شده‌است، در کنفرانس مخابراتی سال 1975 ظاهر شد.

سی‌آرسی یک کد تشخیص خطا است. محاسبه آن شبیه عمل تقسیم اعشاری است که خارج قسمت حذف می‌شود و باقیمانده به عنوان نتیجه در نظر گرفته می‌شود، با این تفاوت مهم که محاسبات آن محاسبات بدون رقم نقلی از یک میدان محدود است. اعلام یک سی‌آرسی خاص با مشخص کردن مقسم و سایر مشخصات آن انجام می‌شود.

اگرچه سی‌آرسی‌ها می‌توانند با استفاده از هر میدان محدودی ساخته شوند، همه سی‌آرسی‌های پرکاربرد از میدان محدود GF(2) بهره می‌برند. این میدانی از دو عنصر، عموما به نام ۰ و ۱، است که به راحتی با معماری کامپیوتر سازگار است. یک دلیل مهم برای محبوبیت سی‌آرسی‌ها برای تشخیص تغییرات تصادفی داده‌ها اطمینان از کیفیت آن‌ها است. نوعا"، یک سی‌آرسی nبیتی، که برای یک بلاک داده با طول دلخواه محاسبه شده‌است، هر حوزه خطای با طول کمتر از n بیت (به عبارت دیگر، هر تغییری که محدوده آن بیش از n بیت مجاور از داده‌ها نباشد) و 1-2^(-n) تعداد از سایر حوزه‌های با طول بیش از n بیت را تشخیص می‌دهد. خطاها در هیچ‌یک از کانال‌های انتقال و رسانه‌های ذخیره‌سازی مغناطیسی دارای توزیع تصادفی نیستند و در نتیجه فایده خواص سی‌آرسی‌ها را نسبت به سایر روش‌های تشخیص خطا از جمله کدهای چندگانه زوجیت بیشتر می‌کنند. ساده‌ترین سامانه تشخیص خطا، بیت زوجیت، در واقع یک سی‌آرسی عادی است که از مقسم دوبیتی ۱۱ استفاده می‌کند.

2-1-5-1 سی‌آرسی‌ها و تمامیت داده‌ها

سی‌آرسی‌ها، به خودی خود، راهکار مناسبی برای حفاظت در مقابل تغییرات عمدی روی داده نیستند (مثلا در برنامه‌های اعتبارسنجی)، چون مبانی ساده ریاضیات آن‌ها باعث می‌شود که بتوان هر تغییر دلخواه را روی داده‌ها طوری اعمال کرد که سی‌آرسی داده‌ها تغییر نکند. اغلب این فرض غلط وجود دارد

که وقتی پیامی به همراه سی‌آرسی آن از یک کانال آزاد دریافت می‌شود و سی‌آرسی دریافتی با سی‌آرسی محاسبه شده مطابقت می‌کند پس پیام ممکن نیست در حین دریافت تغییر کرده باشد. این درست نیست چون هر دوی آن‌ها می‌توانند تغییر کرده باشند، به طوری که سی‌آرسی جدید با پیام جدید مطابقت کند. بنابراین سی‌آرسی‌ها می‌توانند جهت بررسی درستی داده‌ها استفاده شوند ولی نه برای اطمینان از تمامیت آن. ایجاد پیام‌های دیگری که همان سی‌آرسی را ایجاد کنند کار ساده‌ای است، خصوصا پیام‌هایی که بسیار شبیه پیام اصلی هستند. طبق طراحی پیامی که بسیار شبیه پیام اصلی است (و تفاوت آن تنها در یک الگوی تداخل تصادفی است) سی‌آرسی کاملا متفاوتی خواهد داشت و بنابراین تشخیص داده خواهد شد. در مقابل، یک راه موثر برای محافظت پیام‌ها در برابر تغییرات عمدی استفاده از کدهای اعتبار سنجی پیام همچون HMAC است.

2-5-1-2 محاسبه سی‌آرسی

برای محاسبه یک سی‌آرسی دودویی nبیتی، بیت‌های ورودی را در یک سطر بنویسید، و الگوی (n+1)بیتی را که نشان‌دهنده مقسم سی‌آرسی است (و چندجمله‌ای نامیده می‌شود) زیر سمت چپ‌ترین بیت قرار دهید. در زیر، اولین محاسبه برای ایجاد یک سی‌آرسی ۳بیتی نشان داده شده‌است:

```
11010011101100 <--- ورودی

1011 <--- مقسم (4 بیت)
----------------------
01100011101100 <--- نتیجه
```

اگر بیت ورودی بالای سمت چپ‌ترین بیت مقسم صفر باشد، محاسبه‌ای انجام نمی‌شود و مقسم را یک بیت به راست حرکت می‌دهیم. اگر بیت ورودی بالای سمت چپ‌ترین بیت مقسم یک باشد، مقسم و ورودی XOR می‌شوند (به بیان دیگر بیت ورودی بالای هر بیت یک مقسم عکس می‌شود). سپس مقسم را یک بیت به راست حرکت می‌دهیم و این روند تا زمانی تکرار می‌شود که انتهای مقسم به انتهای سطر ورودی نرسیده‌است. در زیر، آخرین محاسبه نشان داده شده‌است:

```
00000000001110 <--- نتیجه محاسبه قبلی
          1011 <--- مقسم
----------------------
00000000000101 <--- باقی‌مانده (3 بیت)
```

از آنجایی که چپ‌ترین بیت مقسم در مواجه با هر بیت یک ورودی آن را صفر می‌کند، وقتی این روند

پایان می‌یابد تنها بیت‌های ورودی که می‌توانند غیر صفر باشند آخرین n بیت سمت راست است. این n بیت، باقی‌مانده مرحله تقسیم است و البته همان مقدار تابع سی‌آرسی است (مگر آنکه تابع سی‌آرسی انتخابی شامل تعدادی پس‌پردازش باشند).

2-1-5-3 مشخصات سی‌آرسی

مفهوم سی‌آرسی به عنوان یک کد تشخیص خطا هنگام پیاده‌سازی آن در یک سامانه واقعی می‌تواند شامل برخی پیچیدگی‌های دیگر نیز باشد. در زیر، تعدادی از آن‌ها آمده‌است:

یک پیاده‌سازی خاص ممکن است یک الگوی بیتی ثابت را پیشوند قرار دهد. این زمانی مفید است که خطاهای ساعتی ممکن است است بیت‌های صفر را در ابتدای پیام قرار دهد و در این صورت با این الگو قابل تشخیص است.

یک پیاده‌سازی خاص ممکن است به پیام n بیت صفر الحاق کند. این می‌تواند بررسی صحت پیامی را که سی‌آرسی به آن الحاق شده‌است ساده‌تر کند. در این روش پس از الحاق n بیت صفر و محاسبه مجدد سی‌آرسی، نتیجه دقیقا صفر می‌شود و باقی‌مانده کافیست با صفر مقایسه شود.

یک پیاده‌سازی خاص ممکن است نتیجه را با یک الگوی ثابت XOR کند.

ترتیب بیت‌ها: برخی روش‌ها کم‌ارزش‌ترین بیت را نخست قرار می‌دهند و برخی بالعکس. ترتیب بیت‌ها در سخت‌افزارهای انتقال سریالی داده بسیار اهمیت دارد زیرا اکثر روش‌های انتقال که به صورت وسیع استفاده می‌شوند از الگوی ابتدا-کم‌ارزش‌ترین-بیت استفاده می‌کنند.

ترتیب بایت‌ها: در سی‌آرسی‌های چند بایتی، ممکن است این تردید پیش آید که آیا بایت منتقل شده اول، کم‌ارزش‌ترین بایت است یا باارزش‌ترین. به عنوان مثال در برخی روش‌ها بایت‌های سی‌آرسی ۱۶بیتی را جابجا می‌کنند.

حذف باارزش‌ترین بیت چندجمله‌ای مقسم: از آنجایی که باارزش‌ترین بیت همیشه یک است، و از آنجایی که یک سی‌آرسی nبیتی باید به صورت یک مقسم (n+1) بیتی تعریف شود و در این صورت می‌تواند از یک ثبات nبیتی سرریز می‌شود، برخی نویسندگان بیان بیت بالای مقسم را غیرضروری می‌دانند.

2-1-5-4 سی‌آرسی‌های پرکاربرد و استاندارد

اگرچه سی‌آرسی‌ها از اجزای معیارها متعددی هستند اما خودشان، از منظر وجود الگوریتمی جهانی، مورد قبول نیستند. به عنوان مثال دو چندجمله‌ای سی‌آرسی-۱۲، ده نوع مستند سی‌آرسی-۱۶ و چهار سی‌آرسی-۳۲ وجود دارد. این چندجمله‌ای‌ها عموما بهترین چندجمله‌ای‌های ممکن نیستند. بین ۱۹۹۳ و ۲۰۰۴، کوپمن، کستاگنولی و سایرین فضای چندجمله‌ای‌ها تا ۱۶ بیت، 24 و ۳۲ بیتی را جهت یافتن مثال‌هایی با کارایی بهتر (از نظر فاصله هامنی برای یک طول پیام خاص) از چندجمله‌ای‌های پروتکل‌های پیشین بررسی کردند و بهترین آن‌ها را در جهت بهبود ظرفیت تشخیص خطای استانده‌های آتی منتشر کردند. به طور خاص، iSCSI یکی از یافته‌های این پژوهش را مورد استفاده قرار داده‌است.

6-1-2 کد گری

نمایش کدهای دودویی که بعد از فرانک گری (Frank Gray) به نام کد گری شناخته شد که یک سیستم از اعداد دودویی است که هر دو عدد متوالی فقط در یک بیت با هم اختلاف داشته باشند. امروزه کدگری به طور گسترده برای تصحیح اشکالات در سیستم ارتباط دیجیتالی مثل کابل‌های تلویزیونی و تلویزیون‌های دیجیتالی جهانی استفاده می‌شود.

یکی از محققان آزمایشگاه بل (Bell) به نام فرانک گری اولین بار به طور رسمی کد گری را مورد استفاده قرار داد و این کد بعد از گری توسط افرادی که از آن استفاده می‌کردند کد گری نامگذاری شد.

1-6-1-2 تاریخچه و کاربردهای علمی

کد گری قبل از آن که در مهندسی به کار رود در جدول‌ها پازل‌های ریاضی به کار برده می‌شد، ریاضیدان فرانسویEmile Boudat از کد گری در سال۱۸۷۸در تلگراف استفاده کرد و برای این کارش مدال دریافت کرد و اما کاربردهای آن، از کد گری به عنوان یک رمزگذار استفاده می‌شود که نسبت به رمزگذار عادی برتری دارد. در نمایش کد گری خاصیت دایره‌ای بودن آن باعث می‌شود که دو عدد دو سر نیز فقط در یک بیت متفاوت باشند. کد گری یک دور همیلتونی در یک مکعب n بعدی Qn تولید می‌کند که هر کدام از اعداد آن یک راس را نشان می‌دهد و نیز در الگوریتم‌های ژنتیکی از آن استفاده می‌شود و نیز البته برچسب گذاری جدول کارنو از موارد دیگر استفاده آن است. زمانی کد گری برای آدرس دهی حافظه در کامپیوتر استفاده می‌شود کامپیوتر نیروی کمتری صرف یافتن آدرس‌ها می‌کند چون هر آدرس با قبلی فقط در یک بیت متفاوت است. طراحان مدارهای منطقی از کد گری به طور گسترده برای عبور چند بیت اطلاعات بین سیستم‌های همزمان استفاده می‌کنند.

دایره کد گری

2-6-1-2 انگیزهٔ پیدایش کد گری

بعضی از دستگاه‌ها وضعیت دستگاه را با کدهای باینری نمایش می‌دهند، اگر این دستگاه‌ها از کد باینری عادی استفاده کند این دو وضعیت پشت سر هم خواهند بود 011 -- > 100 و مشکل کد باینری عادی این است که در حالت طبیعی خیلی بعید نست که چند بیت همزمان تغییر کنند همان طور که در بالا نمایش داده شده‌است که در کد باینری عادی هر سه بیت همزمان تغییر کرده‌اند اما می‌توان اعداد را طوری در کنار هم قرار داد که فقط در یک بیت متفاوت باشند و تغییر زیادی نکنند مثلا" 011 – 001 – 101 – 100 پس کد باینری منعکس شده یا همان کد گری این مشکل را حل می‌کند زیرا که فقط یک بیت در آن‌ها تغییر می‌کند.

	Binary	Gray
0	000	000
1	001	001

2	010	011
3	011	010
4	100	110
5	101	111
6	110	101
7	111	100

با توجه به حالت ۷ و ۰ می‌بینیم که فقط در یک بیت تفاوت دارند که همان خاصیت دوره‌ای یا چرخشی بودن کد گری می‌گوییم.

منابعِ استفاده‌شده برای تهیه‌ی این گزارش:

i. https://blog.faradars.org/error-detection-and-correction/
ii. https://fa.wikipedia.org/wiki/%D8%AA%D8%B4%D8%AE%DB%8C%D8%B5_%D9%88_%D8%AA%D8%B5%D8%AD%DB%8C%D8%AD_%D8%AE%D8%B7%D8%A7
iii. https://www.setakit.com/mag/error-detection-in-computer-networks/
iv. https://www.setakit.com/mag/error-correction/
v. http://www.sahar-it-91.blogfa.com/post/17

پایان.