شبکه های مخابراتی

امیررضا نوروزی

شماره دانشجویی: 404215839

شبیه‌سازی فرآیند CRC و کدینگ Extended Hamming Code

چکیده

در سیستم‌های مخابراتی، تضمین صحت داده‌های منتقل‌شده اهمیت بنیادین دارد؛ زیرا نویز، تداخلات الکترومغناطیسی و خطاهای سخت‌افزاری می‌توانند باعث تغییر محتویات فریم شوند. ازاین‌رو استفاده از روش‌های تشخیص و تصحیح خطا، یکی از اجزای اصلی طراحی پروتکل‌ها و شبکه‌های ارتباطی محسوب می‌شود. در این گزارش، دو روش پرکاربرد در این حوزه یعنی **Cyclic Redundancy Check (CRC)** و **Extended Hamming Code** مورد بررسی، تحلیل و شبیه‌سازی قرار گرفته‌اند. روش CRC با اتکا بر جبر چندجمله‌ای دودویی توانایی بسیار بالایی در تشخیص خطاهای چندبیتی و Burst Errors دارد و به‌عنوان یکی از قابل‌اعتمادترین مکانیزم‌های تشخیص خطا در لایه پیوند داده شناخته می‌شود. در مقابل، Extended Hamming Code با افزودن بیت‌های پاریتی ساختاریافته و یک بیت پاریتی اضافی، امکان **تشخیص خطای چندبیتی** و **تصحیح خطای تک‌بیتی** را فراهم می‌کند و در کاربردهایی که نیازمند خوداصلاحی داده هستند، جایگاه ویژه‌ای دارد.

در این گزارش، شبیه‌سازی دقیقی در محیط Jupyter توسعه داده شده که طی آن، ابتدا داده ورودی با کدینگ Hamming گسترش یافته و سپس CRC بر فریم تولیدشده اعمال شده است. در ادامه، با تزریق خطای کنترل‌شده، کارایی هر یک از روش‌ها و همچنین اثربخشی ترکیبی آنها تحلیل شده است. نتایج به‌دست‌آمده نشان می‌دهد که ادغام این دو روش، ساختاری فراهم می‌کند که ضمن حفظ توانایی تصحیح خطای تک‌بیتی، قادر است خطاهای چندبیتی و خطاهای غیرقابل‌اصلاح را نیز با دقت بالا تشخیص دهد. این ترکیب می‌تواند در سیستم‌های حساس، شبکه‌های صنعتی، مخابرات بی‌سیم و سناریوهای با نرخ خطای بالا، عملکردی پایدار و قابل‌اطمینان ارائه دهد.

1. مقدمه

در سیستم‌های مخابراتی، داده‌ها هنگام ارسال همواره در معرض نویز و اختلال قرار دارند. برای کاهش احتمال خطا، روش‌های مختلف کدینگ استفاده می‌شود که دو مورد از پرکاربردترین آنها **CRC** و **Extended Hamming** هستند.

CRC بیشتر برای **تشخیص خطا** مناسب است، در حالی که Extended Hamming علاوه بر تشخیص، امکان **تصحیح تک‌خطا** را نیز فراهم می‌کند. ترکیب این دو روش در بسیاری از کاربردهای صنعتی و شبکه‌ای موجب افزایش اطمینان انتقال می‌شود.

2.اصول عملکردCRC

CRCبر پایه تقسیم دودویی داده بر یک چندجمله‌ای (Generator Polynomial) عمل می‌کند. خروجی این تقسیم، که همان باقی‌مانده است، به عنوان **چک‌سام** به انتهای داده افزوده می‌شود.
خصوصیت اصلی CRC این است که اگر حتی یک بیت از داده طی انتقال تغییر کند، باقی‌مانده محاسبه‌شده در مقصد با مقدار اصلی مطابقت نخواهد داشت و خطا قابل تشخیص است.

روند کلی کار:

1. انتخاب چندجمله‌ای تولید (برای مثال $G(x) = x^3 + x + 1$) :
2. اضافه کردن صفرهای مورد نیاز به انتهای داده.
3. انجام تقسیم-XOR محور.
4. افزودن باقی‌مانده به داده اصلی و تشکیل فریم نهایی.

3.کدینگExtended Hamming Code

در Hamming معمولی، تعداد بیت‌های افزونه به‌گونه‌ای تعیین می‌شود که بتوان موقعیت یک خطا را پیدا و آن را اصلاح کرد. نسخه Extended یک بیت **Parityکلی** نیز اضافه می‌کند تا امکان تشخیص چند خطا بهبود یابد.

مراحل این کدینگ:

- قرار دادن بیت‌های Parity در موقعیت‌های 2^k
- محاسبه پاریتی هر گروه
- محاسبه parity کلی برای نسخه(Extended)
- تشکیل کد نهایی و ارسال

پس از دریافت، با محاسبه مجدد پاریتی‌ها می‌توان نوع خطا و محل آن را تعیین کرد:

- اگر syndrom≠0 و parity کلی = 1 ⟶ خطای تک‌بیتی
- اگر syndrom≠0 و parity کلی = 0 ⟶ دو خطا یا خطای غیرقابل اصلاح
- اگر syndrom=0 → داده سالم

---

4. ادغام CRC و Extended Hamming Code

در کاربردهای حساس، ابتدا داده با Hamming کد می‌شود تا امکان تصحیح خطا فراهم شود. سپس روی فریم خروجی، CRC اعمال می‌شود تا خطاهای چندگانه نیز قابل تشخیص باشند.
خروجی نهایی، داده‌ای با سطح حفاظت بالا در برابر خطاست.

---

5. شبیه‌سازی در Jupyter Notebook

در شبیه‌سازی معمولاً این مراحل دنبال می‌شود:

1. **تعریف داده ورودی**
2. **پیاده‌سازی تابع CRC** شامل:
   - تقسیم دودویی
   - محاسبه باقی‌مانده
3. **طراحی توابع Extended Hamming** برای:
   - قرار دادن بیت‌های پاریتی
   - تولید syndrome
   - اصلاح خطا
4. **ایجاد خطای مصنوعی** روی یکی از بیت‌ها
5. **بررسی عملکرد هر دو روش** و مقایسه نتایج

در نهایت، با اجرای چند سناریو مشخص می‌شود که:

- Hamming خطای تک‌بیتی را اصلاح می‌کند.
- CRC خطاهای چندبیتی را بهتر تشخیص می‌دهد.
- تلفیق این دو روش پوشش خطایی قوی‌تری نسبت به استفاده تکی از هرکدام فراهم می‌کند.

جمع‌بندی

ترکیب CRC و Extended Hamming یک راهکار مناسب برای طراحی سیستم‌های انتقال داده قابل اعتماد است . Hammingامکان **تصحیح** را فراهم کرده و CRC نوعی **محافظ بیرونی** ایجاد می‌کند که خطاهای پیچیده‌تر را نیز شناسایی می‌کند. شبیه‌سازی این فرآیند در Jupyter به‌خوبی روند تولید بیت‌های افزونه، تشخیص خطا و اصلاح آن را نشان می‌دهد و می‌تواند مبنایی برای تحلیل دقیق‌تر در پروژه‌های مخابراتی باشد.
