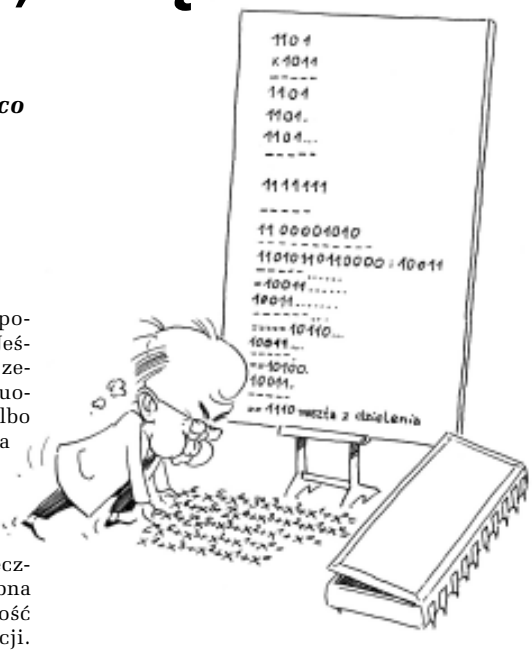
CRC doda Ci pewności, część 1

O tym, że istnieją metody zabezpieczania transmisji wiedzą z pewnością wszyscy nasi Czytelnicy. Niewielu wie natomiast, co tak naprawdę kryje się pod magicznym pojęciem „CRC“, które jest często traktowane jako synonim bezpieczeństwa wymiany danych. Tajemnice CRC wyjaśnimy w cyklu artykułów, którego część pierwszą publikujemy w tym właśnie numerze EP.



Chyba większość Czytelników spotkała się kiedyś z tytułowym określeniem - CRC. Co to jest? Na ogół CRC jest kojarzone z zagadnieniami związanymi z transmisją danych. Niektórzy potrafią rozszyfrować ów akronim (*Cyclic Redundancy Codes* lub *Check*), a zupełnie nieliczni wiedzą dokładnie o co chodzi. Nic dziwnego. Powiązania z niełatwym aparatem matematycznym skłaniają do obchodzenia problemu szerokim łukiem. Proponuję jednak chwilę cierpliwości. Okazuje się, że „nie taki diabeł straszny jak go malują“.

Problemy podstawowe

Czy elementowa stopa błędów transmisji danych równa 10^{-12} to dużo, czy mało? Parametr ten oznacza, że na bilion wysłanych elementów, statystycznie jeden może zostać przekłamany. Dla lepszego uświadomienia sobie proporcji załóżmy, że na Ziemi żyje 4000 mln ludzi. Liczba 10^{-12} odpowiada więc jednemu człowiekowi wybranemu spośród mieszkańców 250 planet takich jak Ziemia. Pytanie pozostaje aktualne. Dużo to, czy mało? Hmmm, wydaje się że mało, nawet bardzo mało. Wróćmy więc do transmisji danych charakteryzującej się taką właśnie elementową stopą błędów. Nie jest to wartość abstrakcyjna. Parametrem takim charakteryzuje się np. system 10 Gbd Ethernet. Strach pomyśleć o skutkach ewentualnych błędów, gdyby przesyłane dane dotyczyły np. systemów podtrzymania życia czy operacji finansowych. Jeśli mówimy o przepływności toru transmisyjnego równego np. 10 Gbd, to dla wyżej założonej elementowej stopy błędów możemy oczekiwać przekłamań co 100 sekund. Spodziewany co niespełna 2 minuty błąd nie może pozostawiać nas w spokoju. Zdecydowanie trzeba takim sytuacjom jakoś zaradzić. Jedną z metod wyeliminowania skutków błędów transmisji jest kontrola poprawności jej przebiegu. W praktyce może wyglądać tak: wysyłane dane grupowane są w bloki, do każdego bloku dołączana jest na końcu dodatkowa, krótka informacja zawierająca specyficzny rodzaj „streszczenia“ zawartości bloku. Odbiornik odbiera komplet informacji, dokonuje samodzielnego streszczenia bloku, po czym sprawdza, czy własna wersja streszczenia odpowiada wersji odebranej. W praktyce streszczanie odbieranego bloku jest najczęściej prowadzone bezpośrednio w czasie transmisji (*on-line*). Jeśli porównanie jest pomyślne (streszczenia są identyczne), praca jest kontynuowana, jeśli niecały odebrany blok albo jest ignorowany (jeśli możemy sobie na to pozwolić), lub wysłane jest do nadajnika żądanie ponownej transmisji. W pewnych sytuacjach możliwe jest nawet samodzielne skorygowanie treści po stronie odbiorczej, bez konieczności powtarzania transmisji. Potrzebna jest do tego jednak pewna nadmiarowość (redundancja) przesyłanych informacji. Warto w tym miejscu zaznaczyć, że CRC może być stosowane nie tylko do kontroli transmisji danych. Za pomocą CRC można również kontrolować np. poprawność zapisu danych na nośniku magnetycznym. Użytkownicy popularnych programów archiwizujących, jak np. ZIP lub RAR również korzystają z jego usług. O metodach obliczania CRC będzie mowa w dalszej części artykułu.

Wykrywanie błędów

Świat nie jest idealny, o tym dobrze wiemy. Człowiek zawsze dążył i dąży do tego stanu, ale powiedzmy sobie szczerze, szanse są raczej marne. Dlatego, póki co, musimy sobie jakoś radzić w trudnych sytuacjach. W naszych rozważaniach będziemy się trzymać zagadnień transmisji danych. Głównym jej wrogiem są szumy kanału transmisyjnego i jego zakłócanie. To właśnie one powodują, że odebrana informacja niekoniecznie będzie zgodna z oryginałem, co w pewnych sytuacjach może mieć bardzo, bardzo przykre skutki. Środkiem na to, by móc wykryć ewentualne błędy jest - jak już doszliśmy wcześniej - dodanie pewnego elementu do przekazywanych danych. Informację taką nazywamy sumą kontrolną, chociaż sumowanie nie zawsze musi być rozumiane dosłownie, przynajmniej w znaczeniu do jakiego przywykliśmy. Można nawet powiedzieć więcej, im będzie ono bardziej wymyślne, tym większą będzie miało skuteczność. Na początku pozostańmy jednak przy dosłownym rozumieniu tego słowa, z zastrzeżeniem, że dotyczy ono liczb mniejszych od 256. Mówimy o sumie modulo 256. Wyobraźmy sobie, że mamy do czynienia z transmisją przedstawioną poniżej (wszystkie liczby zapisane dziesiętnie):
wiadomość: **6 23 4**
wiadomość z sumą kontrolną: **6 23 4 33**
odebrana wiadomość: **6 27 4 33**

W powyższym przykładzie dopisana na końcu suma kontrolna jest utworzona poprzez zsumowanie (modulo 256) wszystkich elementów bloku danych. W odebranej wiadomości nastąpiło przekłamanie drugiego elementu. Zamiast 23 odebrano 27. Wyliczona w odbiorniku suma kontrolna odebranego bloku nie jest zgodna z przesłaną sumą kontrolną (6+27+4=37≠33), co może być podstawą do stwierdzenia, że odebrany blok nie jest identyczny z oryginałem. Nie jest to jednak mechanizm gwarantujący wystarczającą pewność kwalifikacji odbieranych danych. Np. pojedynczy błąd odebranej sumy kontrolnej może być powodem uznania bloku danych jako błędnego, mimo że jest on prawidłowy. Problem powiększa się, jeśli dopuścimy błędy wielokrotne, nie mówiąc już o drastycznym zmniejszeniu skuteczności metody, gdy zwiększymy długość bloku danych, pozostawiając nadal sumowanie modulo 256. W powyższym przykładzie prawdopodobieństwo niewykrycia błędów wielokrotnych jest równe 1:256. Błędy wielokrotne mogą na tyle zafałszować wynik, że nie zauważymy ich, nawet jeśli faktycznie wystąpią, jak choćby w poniższym przykładzie:
wiadomość: **6 23 4**
wiadomość z sumą kontrolną: **6 23 4 33**
odebrana wiadomość: **8 20 5 33**

Sumy kontrolne zgadzają się, ale blok odebrany nie jest identyczny z nadawanym. Sposobem na poprawienie skuteczności sprawdzania poprawności transmisji może być zastosowanie dłuższego rejestru służącego do tworzenia sumy kontrolnej. Jeśli zastosujemy rejestr 16-bitowy (mówimy więc o sumowaniu modulo 65536), znacznie zredukujemy prawdopodobieństwo niewykrycia błędu. W konkretnym przypadku zmaleje ono do wartości 1:65536. Jednakże główną przyczyną niepowodzeń jest mała sku-

Bezpieczna wymiana danych w systemach mikroprocesorowych

teczność samej metody. Możemy powiedzieć, że zastosowany sposób obliczania sumy kontrolnej, jako zwykłej sumy modulo N, jest za mało „przypadkowy". Każdy nadchodzący bajt oddziaływuje jedynie na jeden bajt rejestru sumującego bez względu na jego całkowitą długość. Z tego powodu nie jest możliwe wykrycie błędu jaki wystąpił w drugim przykładzie.

Skutecznym rozwiązaniem może być więc jedynie zastosowanie bardziej wyszukanej metody prowadzenia obliczeń, zapewniającej operowanie na całej szerokości rejestru sumacyjnego. Widzimy więc dwa aspekty opracowania metody obliczania sumy kontrolnej. Po pierwsze: dobranie odpowiedniej długości rejestru sumacyjnego, zapewniającego odpowiednio niskie prawdopodobieństwo błędu (np. rejestr 32-bitowy daje prawdopodobieństwo błędu $1/2^{32}$). Po drugie: zastosowanie takiej formuły obliczeniowej, która zapewni potencjalną zmianę dowolnego bitu w całym rejestrze sumacyjnym dla dowolnego bajtu wejściowego. Metody takie oczywiście opracowano i będziemy o nich dalej mówić. Stosowane do nich określenie *checksum* (suma kontrolna) ma raczej historyczne znaczenie. Dzisiaj czyste sumowanie zostało zastąpione operacjami bardziej złożonymi.

Podstawowe zasady obliczeń CRC

Obecnie stosowane metody właściwie bardziej przypominają dzielenie niż sumowanie, cały czas jednak mówimy o sumowaniu. Ach te przyzwyczajenia! Ale przecież dzielenie to wielokrotne odejmowanie, a odejmowanie to dodawanie ze zmienionym znakiem. Tym nieco pokrętnym rozumowaniem usprawiedliwiliśmy stosowaną nomenklaturę, więc spokojnie możemy przystąpić do dalszego zgłębiania tematu. Usprawiedliwienie jest tym bardziej stosowne, że później w praktyce, gdy przyjdzie nam tworzyć konkretne programy dla mikroprocesorów, będziemy korzystać z elementarnych rozkazów dodawania i odejmowania.

Zachowując łączność z wcześniejszymi rozważaniami przyjmiemy, że długość rejestru sumacyjnego będzie odpowiadała długości dzielnika w nowej idei. Teraz transmitowaną wiadomość traktujemy jako ogromną (w ogólnym przypadku) liczbę binarną, która będzie dzielona przez liczbę o ustalonej długości. Reszta z dzielenia będzie naszą sumą kontrolną. Dalsze postępowanie jest takie samo, jak wcześniej. Odbiornik odbiera wiadomość, wykonuje dzielenie, bierze jego resztę traktując jako CRC i porównuje z wartością odebraną. Przykładowo: wiadomość zawiera dwa bajty 6 i 23 (jak w pierwszym przykładzie). W zapisie szesnastkowym będą to wartości 06h 17h, a rozpisane binarnie dadzą: 00000110b 00010111b. Przyjmijmy jako dzielnik jednobajtową liczbę równą 00001001b. Tak więc CRC otrzymamy jako resztę z dzielenia liczby 0000011000010111b przez 1001b. Obliczenie wykonamy znaną ze szkoły metodą pisemną z tym, że będziemy je wykonywać na liczbach binarnych. Przypomnijmy krótko zasady na poniższym przykładzie:

```
 ''
 11010
-01100
 -----
 01110 (odejmowanie liczb binarnych)
```

Odejmujemy od prawej strony: 0-0=0; 1-0=1; 0-1=1 (1 pożyczamy z sąsiedniej, lewej kolumny). W następnej kolumnie mamy 1-1, ale 1 musieliśmy pożyczyć, mamy więc 0-1=1 (1 znowu pożyczamy z kolejnej kolumny; w ostatniej mielibyśmy 1-0, ale podobnie jak wcześniej 1 już pożyczyliśmy, więc ostatecznie jest 0-0=0. Sprawdźmy na liczbach dziesiętnych. 11010b=26 (przyp. literka b oznacza, że chodzi o liczbę binarną), 01100b=12, 01110b=14. 26-12=14. Zgadza się, wracamy zatem do obliczania CRC (kropki ułatwią podpisywanie cyfr w odpowiednich miejscach):

```
      10101101
----------------
0000011000010111: 1001
    1001.......
    ---.....
    ==1100.....
      1001.....
      ---...
      ==1110...
        1001...
        --..
        =1011..
         1001..
         ---
         ==1011
           1001
           --
           ==10= 2 =
              = reszta z dzielenia
```

Sprawdźmy dziesiętnie: 1559:9=173 reszta 2. Powyżej otrzymaliśmy w wyniku liczbę 10101101b=173 i resztę 2. Obliczenie jest zatem prawidłowe. Zauważmy, że przekłamanie dowolnego bitu wiadomości może wpłynąć na ostateczną wartość całej reszty z dzielenia. Tak nie było w przypadku zwykłego dodawania. Dlatego powyższa metoda wykazuje dużo większą skuteczność wykrywania błędów.

Arytmetyka wielomianów

Rezultat uzyskany w poprzednim przykładzie stanowi znaczny krok do przodu w porównaniu z pierwszym pomysłem. Daleko nam jednak do doskonałości. Przyjmijmy to na razie na wiarę. Komplikujemy więc metodę, w nadziei na osiągnięcie kolejnej poprawy. Czeka nas wcześniej jeszcze jedna porcja teorii. Każdy, kto przebrnął przez szkołę średnią, na pewno miał do czynienia z wielomianami i zadawał sobie wtedy pytanie po co uczy się takich bzdur. Tym, których dopiero to czeka, radzę jednak nie spać na lekcji. Arytmetyka wielomianów jest bowiem podstawowym narzędziem wykorzystywanym w teorii kodowania nadmiarowego. Omawiane wcześniej składniki operacji dzielenia, czyli dzielna, dzielnik i iloraz reprezentują dodatnie liczby całkowite. Każda taka liczba jest ciągiem bitowym, którego poszczególne bity stanowią współczynniki wielomianu (binarne). Dla lepszego zrozumienia tematu wróćmy do pierwszego przykładu. Mieliśmy tam daną równą 23=17h=10111b. Odpowiada jej więc wielomian:

$1*x^4+0*x^3+1*x^2+1*x^1+1*x^0$

lub krócej:

$x^4+x^2+x^1+x^0$

Przypuśćmy teraz, że chcemy pomnożyć dwie liczby 1101b i 1011b. Zastosujemy oczywiście mnożenie wielomianów (pamiętamy ze szkoły jak to się robi):

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

Niepokój może budzić składnik $3*x^3$. Jeśli przyjmiemy, że x=2, okaże się, że składnik ten generuje przeniesienie na sąsiednią, starszą pozycję. Mamy więc:

$x^6 + x^5 + x^4 + 3*x^3 + x^2 + x^1 + x^0 = x^6 + x^5 + 2*x^4 + x^3 + x^2 + x^1 + x^0 = x^6 + 2*x^5 + x^3 + x^2 + x^1 + x^0 = 2*x^6 + x^3 + x^2 + x^1 + x^0 = x^7 + x^3 + x^2 + x^1 + x^0$

Gdybyśmy nie znali wartości x, przeniesienie takie nie byłoby możliwe, ponieważ nie moglibyśmy stwierdzić, czy $3*x^3$ jest równe $x^4 + x^3$, czy też nie. Odrzucając więc przeniesienia, wynik powyższego mnożenia byłby równy:

$x^6 + x^5 + x^4 + x^3 + x^2 + x^1 + x^0$

Tu uwaga: powyższy wynik nie będzie oczywiście prawidłowy w klasycznej arytmetyce. Rezygnację z przeniesień wprowadzamy świadomie, zdając sobie z tego sprawę. Czy takie podejście może być przydatne do kalkulowania CRC? Popróbujmy.

Arytmetyka binarna bez przeniesień

Przyjmijmy, że dodawanie dwóch liczb w arytmetyce CRC jest identyczne ze zwykłą arytmetyką liczb binarnych z wyjątkiem niestosowania przeniesień. Oznacza to, że każda para odpowiednich bitów określa tylko bit wyjściowy na tej samej pozycji, bez wpływu na pozostałe pozycje. Zilustrujmy to przykładem dodawania, dla którego operacje elementarne to:

```
0+0=0
0+1=1
1+0=1
1+1=0 (brak przeniesienia)
```

Dodawanie liczby binarnej przebiega więc następująco:

```
 10011011
+11001010
---------
 01010001
```

Podobnie jest dla odejmowania, dla którego operacje elementarne przedstawiono poniżej:

```
0-0=0
0-1=1 (bez pożyczki)
1-0=1
1-1=0
```

Samo odejmowanie wygląda następująco:

```
 10011011
-11001010
---------
 01010001
```

Zauważmy, że elementarne działania (dodawanie i odejmowanie) dają ten sam wynik dla analogicznych par argumentów. Można je więc zastąpić jednym działaniem odpowiadającym operacji XOR. Od tej chwili będziemy ją oznaczać symbolem ⊕. XOR można w związku z powyższym spostrzeżeniem uznać za działanie odwracalne. Takie podejście ułatwia nieco obliczenia, pozwala nam stosować tylko jeden ro-

dzaj operacji, prowadzi jednak do sytuacji, które mogą się wydać nawet absurdalne w ujęciu arytmetyki klasycznej. Dla przykładu rozpatrzmy, która z liczb będzie większa 1010b czy 10b. Intuicyjnie czujemy, że 1010b, bo ma przecież więcej cyfr znaczących. To samo pytanie postawione w stosunku do liczb 1010b i 1001b nie da już jednoznacznej odpowiedzi, bo okazuje się, że jedną z nich można uzyskać zarówno poprzez dodanie, jak i odjęcie pewnej liczby:

```
1001b = 1010b + 0011b i 1001b = 1010b
- 0011b
```

Wynik nonsensowny w arytmetyce klasycznej. Popatrzmy teraz jak będzie wyglądało mnożenie w arytmetyce CRC:

```
   1101
  x1011
  -----
   1101
  1101.
 1101...
 -------
 1111111
```

Pozostało do rozpatrzenia jeszcze dzielenie. Tu sprawa jest nieco bardziej skomplikowana, ponieważ musimy umieć określić, czy dzielnik mieści się we fragmencie dzielnej (przypomnijmy sobie dzielenie pisemne) rozpatrywanym w danym kroku obliczeniowym. To z kolei wiąże się z określeniem, czy dzielnik jest mniejszy od tego fragmentu. Kilka linijek wyżej mieliśmy z tym pewne problemy. Przyjmujemy więc definicję: X jest większe lub równe Y, jeśli pozycja najstarszego bitu równego 1 w liczbie X jest większa lub taka sama jak najstarszego bitu równego 1 w liczbie Y. A oto przykład dzielenia:

```
   1100001010
--------------
11010110110000: 10011
10011.........
----.........
=10011........
 10011........
 -----...
 =====10110...
     10011...
     ---.
     ==10100.
       10011.
       ---
       ==1110  reszta z dzielenia
```

Do rozpatrzenia pozostaje jeszcze określenie dwóch zależności między liczbami. Chodzi mianowicie o określenie, czy liczba A stanowi wielokrotność lub podwielokrotność liczby B. Jeśli A jest w arytmetyce CRC wielokrotnością liczby B, to da się ją przedstawić jako zero XOR-owane z odpowiednimi przesunięciami liczby B. Zrozumiemy to, gdy przeanalizujemy poniższy przykład. Niech A=0111010110b, a B=11b. Zgodnie z powyższym A możemy skonstruować następująco:

```
 0000000000
 ⊕.......11.
 ⊕....11....
 ⊕...11.....
 ⊕.11.......
 -----------
 0111010110 = A (znak ⊕ oznacza
operację XOR)
```

Jakakolwiek drobna zmiana liczby A spowoduje, że nie da się stworzyć opisywanej wyżej konstrukcji. Jako zadanie domowe proponuję sprawdzić, czy A=0111010111b jest podzielne przez liczbę B=11b (w rozumieniu arytmetyki CRC). Przypominam, że odpowiedź jest twierdząca, jeśli możliwe jest uzyskanie liczby A jako sumy modulo 2 (XOR) przesunięć liczby B. Ostatecznym sprawdzeniem będzie wykonanie dzielenia A:B dla obu powyższych przykładów ze szczególnym zwróceniem uwagi na resztę z dzielenia.

Teorii było sporo. Trzeba ją teraz jeszcze raz przeanalizować. W następnym odcinku spróbujemy przybliżyć się do praktyki.

Jarosław Doliński, AVT
jaroslaw.dolinski@ep.com.pl

Artykuł powstał na podstawie publikacji „A painless guide to CRC error detection algorithms", autor Ross N. Williams. Można go znaleźć pod adresem http://www.riccibitti.com/crcguide.htm.