Sieci Komputerowe

Mirosław Skrzewski
mskrzewski@polsl.pl

Zapis wielomianowy ciągów kodowych

Ciąg kodowy może być traktowany jako ciąg współczynników stojących przy kolejnych potęgach wielomianu fikcyjnej zmiennej x

$$a(x) = a_0 + a_1 x + a_2 x^2 + \ldots + a_{k-1} x^{k-1}$$

odpowiednik k-bitowego ciągu

Na wielomianach można zdefiniować operacje dodawania, mnożenia i dzielenia, posługując się arytmetyka modulo dwa.

Mnożenie wielomianów:

$$a(x) = a_0 + a_1 x + a_2 x^2 + \ldots + a_{k-1} x^{k-1}$$

$$b(x) = b_0 + b_1 x + \ldots + b_{l-1} x^{l-1}$$

$$c(x) = a(x) * b(x) = c_0 + c_1 x + \ldots + c_{k+l-2} x^{k+l-2}$$

$$c_i = a_0 * b_i + a_1 * b_{i-1} + \ldots + a_i * b_0$$

Dzielenie wielomianu przez wielomian

$$\frac{a(x)}{b(x)} = \frac{a_{k-1}x^{k-1} + a_{k-2}x^{k-2} ... + a_1 x + a_0}{b_{l-1}x^{l-1} + b_{l-2}x^{l-2} + ... + b_1 x^1 + b_0} = c(x); \quad r(x)$$

k > l; c(x) – wynik, stopnia (k-l); r(x) – reszta z dzielenia stopnia < (l-1)

$$a(x) = b(x) * c(x) + r(x)$$

Wielomian *a(x)* dla którego *r(x)* $\equiv$ *0* nazywamy podzielnym przez wielomian *b(x)*.

Podzielność przez wielomian pozwala podzielić zbiór ciągów k-bitowych na ciągi poprawne danego kodu i ciągi niepoprawne (z r(x) ≠ 0)

Kody zbudowane na tej zasadzie (podzielność przez wielomian g(x)) nazywane są *kodami ilorazowymi*.

Podgrupą kodów ilorazowych są *kody cykliczne*. (takie, w których przez cykliczna rotację jednego ciągu poprawnego kodu otrzymuje się wszystkie pozostałe).

$$A(x) = a_{n-1}x^{n-1} + a_{n-2}x^{n-2} + \ldots + a_1 x + a_0$$

Cykliczne przesunięcie elementów ciągu *A(x)* w lewo odpowiada pomnożeniu wielomianu *A(x)* przez *x* i uzupełnieniu wyrazu wolnego.

$$A'(x) = x * A(x) = a_{n-2}x^{n-1} + a_{n-3}x^{n-2} + \ldots + a_1 x^2 + a_0 x + a_{n-1} - a_{n-1} + a_{n-1}x^n$$

$$A'(x) = x * A(x) + (a_{n-1}x^n + a_{n-1})$$

Jeżeli A(x) jest poprawnym ciągiem kodu cyklicznego, podzielnym przez wielomian g(x), to x*A(x) powinien być także podzielny przez g(x).

$$\frac{A'(x)}{g(x)} = \frac{x * A(x) + a_{n-1}(x^n + 1)}{g(x)}$$

a stąd wynika, że $(x^n + 1)$ musi być podzielny przez wielomian g(x), nazywany *generatorem kodu.*

Wniosek - g(x) musi być jednym z wielomianów nierozkładalnych niższych stopni na które można rozłożyć wielomian $(x^n + 1)$

To pozwala nam wyszukać odpowiedni generator kodu dla ciągu o długości n bitów.

Dla wybranych długości ciągów kodowych $N = 2^{k-1}$ wykazano, że $x^N + 1$ da się przedstawić jako iloczyn wielomianów *nierozkładalnych* stopni będących podzielnikami k.

Przykładowe długości ciągów kodowych N = 7, 15, 31, 63, ….

N = 15 = $2^4 - 1$ ma podzielniki 1, 2, 4 więc kandydatów na wielomian generacyjny *g*(x) kodu o długości 15 należy szukać wśród wielomianów nierozkładalnych stopnia 4.

Dalej obowiązuje równanie $2^{(n-k)} - 1 \geq C_1^n = n$, czyli (n-k) powinno wynosić 4 (potrzeba 15 + 1 kombinacji dla identyfikacji ciągów błędów)

Założenia kodowania: odbieramy ciąg *ξ(x) = s(x) + c(x)*

Sprawdzenie odebranego ciągu polega na podzieleniu przez generator g(x)

$$\frac{\xi(x)}{g(x)} = \frac{s(x) + c(x)}{g(x)} = \frac{c(x)}{g(x)} = r(x)$$

Wielomian $x^N + 1$ można rozłożyć na wielomiany stopnia 1 (1), 2 (1) i 4 (3).

$$x^N + 1 = (x+1)(x^2 + x + 1)(x^4 + x^3 + x^2 + x + 1)(x^4 + x^3 + 1)(x^4 + x + 1)$$

Wielomian stopnia 4-ego daje reszty z dzielenie 4-bitowe, wiec może być dobrym kandydatem na wielomian generacyjny g(x).

$x^N + 1$ ma 3 takie wielomiany: $(x^4 + x^3 + x^2 + x + 1)$, $(x^4 + x^3 + 1)$, $(x^4 + x + 1)$

Sprawdzenie kandydata wymaga sprawdzenia, ile różnych reszt otrzymamy dzieląc poszczególne ciągi błędów przez danego kandydata na wielomian generacyjny , czyli wykonując dzielenia c(i) : g(x).

```
00000 00000 00001
00000 00000 00010
00000 00000 00100
.  .  .
```

Kolejne ciągi błędów dla pojedynczych przekłamań to zbiór zer z wędrującą jedynką.

Ciąg $x^4 + x^3 + x^2 + x + 1$ to 11111

```
000000000000001:11111
           11111
          .  .  .
           10000
           11111
          =11110
            11111
           =0001
```

```
0001
0010
0100
1000
1111
0001
```

Dla pierwszych 4 ciągów błędów nie można wykonać dzielenia -> c(i) wprost są resztami z dzielenia.
Dla przekłamania na 6 bicie dostajemy ponownie resztę 00001 - jest 5 różnych reszt r(x). – odpada.

Wielomian $x^4+x^3+x^2+x+1$ generuje tylko 5 różnych reszt, bo wchodzi także w rozwinięcie wielomianu x^5+1

Sprawdzamy kolejny wielomian - x^4+x^3+1 to 11001

```
0..000001:11001
          10000
          11001
          =10010
            11001
            =10110
              11001
              =11110
                11001
                =011100
                   11001
                   =010100
                      11001
                      =11010
                        11001
                        =0011000
                            11001
                            =0001
```

Tablica syndromów kodu

reszta	nr bitu przekłamanego
0001	1
0010	2
0100	3
1000	4
1001	5
1011	6
1111	7
0111	8
1110	9
0101	10
1010	11
1101	12
0011	13
0110	14
1100	15

$g(x) = x^4 + x^3 + 1$ daje 15 różnych reszt z dzielenia, może być generatorem kodu cyklicznego.

Kodowanie informacji do przesyłu.

Poprawne ciągi kodu, podzielne przez wielomian generacyjny g(x) można otrzymać mnożąc wielomian informacyjny a(x) * g(x).

Np. a(x) = 10110110   g(x) = 11001

s(x):

```
10110110 * 11001
        10110110
     10110110
    10110110
----------------
    111001100110
```

Kodowanie nierozdzielne – informacja zmieszana z bitami kontrolnymi

Odbiór informacji

Sprawdzenie informacji odebranej – dzielimy odebrany ciąg przez wielomian generacyjny kodu g(x)

```
    10111001
 111011100110: 11001
 11001
 ==10011
   11001
   =10100
    11001
    =11010
     11001
     ===11110
        11001
        =0111
```

Otrzymaliśmy resztę z dzielenia różną od zera -> wystąpił błąd

Z tablicy syndromów kodu 0111 odpowiada przekłamaniu 8 bitu

Po korekcji (zmianie bitu z 1 na 0) dla odebrania poprawnej informacji należy *powtórzyć proces odbioru.*

Odebraną informacją jest wynik dzielenia ciągu przez g(x) przy reszcie z dzielenia równej 0.

Sprawdzenie po korekcji przekłamania

```
    ........
111011100110        Ciąg odebrany przed korekcją
    ↓
    10110110                   Po korekcji
____________
111001100110:11001             Odbiór informacji
11001
==10111
  11001
  =11100
   11001
   ==10101
     11001
     =11001
      11001
      =====0
```

Kodowanie rozdzielne (*rozdzielnopozycyjne*)

Wymagany efekt – ciąg wysyłany s(x) podzielny przez wielomian g(x) można otrzymać także w inny sposób:

- Mnożymy wielomian a(x) przez x^k (uzupełniamy a(x) ciągiem k zer)
- Dzielimy uzupełniony zerami a(x) przez g(x)
- Otrzymana resztę z dzielenia (to, co „zabrakło" do podzielności) dopisujemy do a(x) w miejsce ciągu zer.

$$s(x) = a(x)^* x^k + r(x)$$

a(x)	r(x)
dane	CRC

a(x) = 10110110   g(x) = 11001 x^k = 10000

a(x)* x^k = 10110110 0000

```
 101101100000:11001
 11001
 =11111
  11001
  ==11010
    11001
    ===11000
       11001
       ==0010  <- r(x)
```

s(x) = 101101100010

Sprawdzenie przy 2 przekłamaniach

101101100010

```
101001101010:11001
11001
=11011
 11001
 ===10101
    11001
    =11000
     11001
     ===0110
```

reszta r(x)

Tablica syndromów

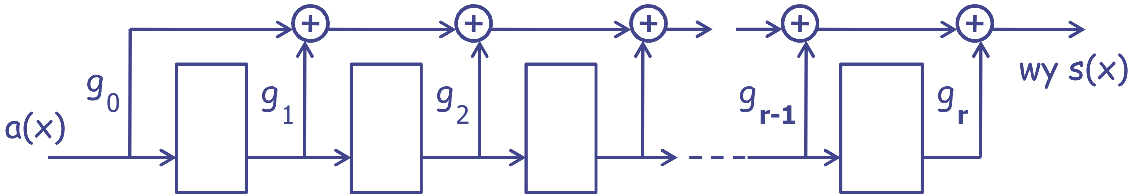
0001	1
0010	2
0100	3
1000	4
1001	5
1011	6
1111	7
0111	8
1110	9
0101	10
1010	11
1101	12
0011	13
0110	14
1100	15

Bit 14 nie istnieje, łatwo stwierdzić, że było > 1 przekłamanie, ale często nie jest to takie jasne.

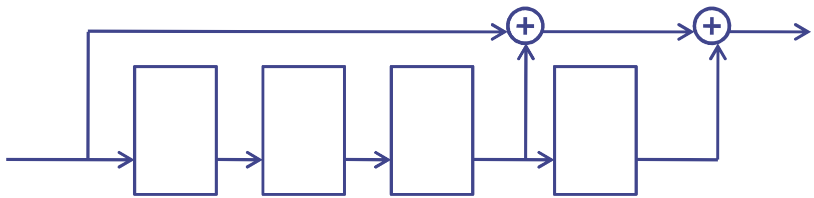
Realizacja sprzętowa

Mnożenie dowolnego wielomianu przez wielomian o stałej postaci można zrealizować z wykorzystaniem rejestrów przesuwnych.

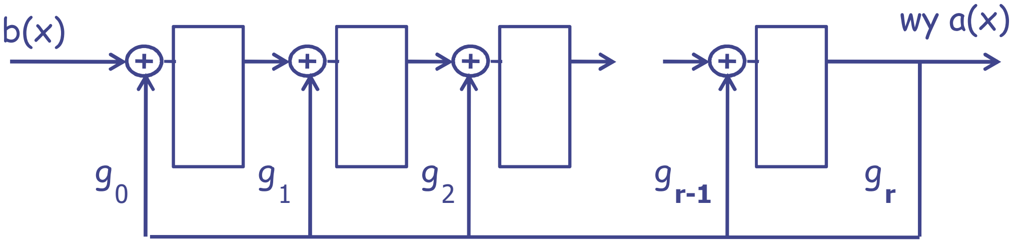
Mnożenie dowolnego a(x) przez stały g(x) stopnia r



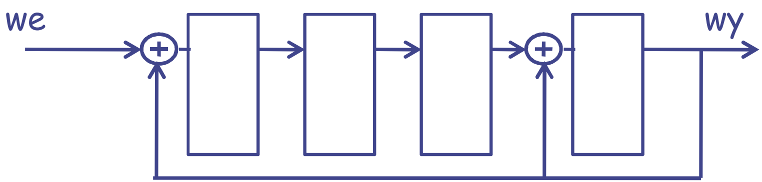
g(x) = $x^4 + x^3 + 1$



Dzielenie wielomianu b(x) przez stały wielomian g(x)



g(x) = $x^4 + x^3 + 1$

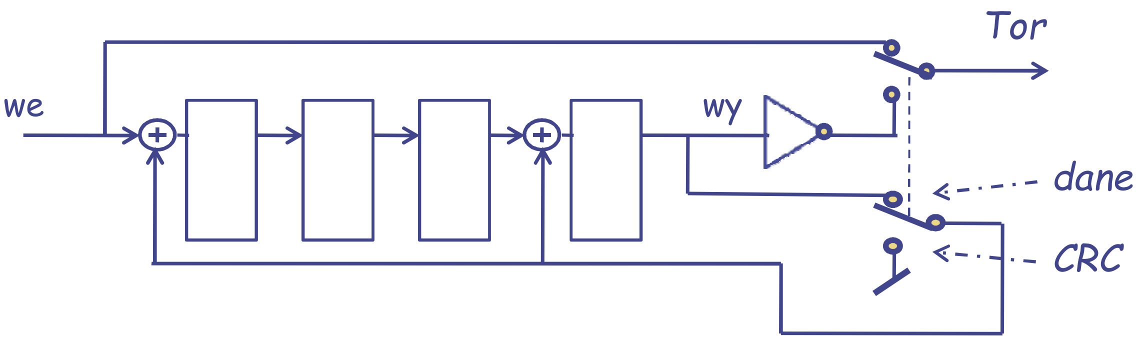


Rozwiązanie transmisji informacji – problemy:

Przerzutniki rejestru przesuwnego są zerowane przed transmisją – rejestr nie reaguje na początkowe zera w informacji.

Wymuszenie reakcji na zera wymaga presetu przerzutników na jedynkę – zmienia się model działania algorytmu. Stan rejestru po udanej transmisji jest różny od zera – stała kombinacja bitów.

Układ nadawania



Generatory kodów cyklicznych

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

Sieci WAN

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

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

Sieci LAN

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

Korekcja błędów większej krotności

„Metoda" Hamminga

1. Dla ciągu k bitów informacji tworzymy kod (*n*, *k*) – dodajemy n-k pozycji kontrolnych – 1-błąd
2. Traktując n jako ciąg informacyjny tworzymy kod (*n1*, *n*) dodając n1 – n pozycji kontrolnych dla korekcji 2-go błędu
3. . . . (*n2*, *n1*) . . .

Łatwo przekonać się, że jest to teoretyczny pomysł, nie bardzo sprawdzający się w praktyce, pomijając sprawność kodowania.

Korekcja błędów grupowych

Błędy występujące na sąsiednich bitach (paczka błędów).

Można zaproponować metodę korygującą paczki o określonej max. długości korzystając z kodów korygujących 1 przekłamanie.

Przy przesyle blokami np. po 16 bitów blok zapisujemy jako 4 wiersze po 4 bity. Każdy wiersz kodujemy np. kodem Hamminga, dodając 3 pozycje kontrolne (razem wiersze 7 bitowe).

Utworzoną macierz wysyłamy kolumnami. Odbierany ciąg bitów zapisujemy analogicznie kolumnami. Pojedyncze przekłamanie grupy do 4 sąsiednich bitów rozłoży się po jednym bicie w każdym wierszu, co zostanie poprawnie skorygowane.

Metoda nie skutkuje przy wystąpieniu 2 lub więcej niezależnych błędów (nie obejmujących sąsiednich bitów).

Kody korygujące 2 i > przekłamań

Wielomiany arytmetyczne można przedstawić w postaci iloczynu jednomianów z pierwiastkami wielomianu typu

z(x) = (x – a)*(x-b)*(x-c)*(x-d)   (dla wielomianu 4 stopnia.)

Zastosowanie tego pomysłu do ciągów binarnych prowadzi do projektu kodów BCH (Bosego, Ray-Chaudhuri, Hocquengheim) i ciał Galois GF. (podklasy kodów cyklicznych).

Generator kodu BCH jest iloczynem generatora kodu cyklicznego g(x) i pewnego wielomianu m(x), o postaci zależnej od ilości korygowanych błędów.

Dla korekcji 2 błędów $g'(x) = (x^4 + x + 1)(x^4 + x^3 + x^2 + x + 1)$ daje

kod (15,7) z 8 pozycjami kontrolnymi, generatorem kodu

$g'(x) = x^8 + x^7 + x^6 + x^4 + 1$ i odległością Hamminga d >= 5

Podklasą BCH są kody Reeda-Solomona (2 błędy), kody Golay'a (3 błędy)

Wykorzystanie kodów detekcyjnych / korekcyjnych

Każdy z kodów, niezależnie od stosowanego rozwiązania ma:

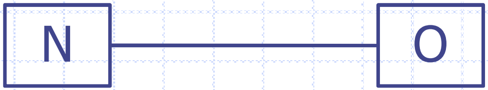
- swoją pulę błędów niewykrywalnych
- własności korekcji kodu zależne są od prawdopodobieństwa występowania krotności błędów większych od obsługiwanej przez daną metodę kodowania.

Korzystanie wyłącznie z możliwości poprawy błędów przez kod jest rzadko wykorzystywane w praktyce. Częściej wykorzystuje się kody korekcyjne jako „silniejsze” kody wykrywające przekłamania, a korekcję realizuje się poprzez retransmisję informacji.

Urządzenia cyfrowe przy wymianie informacji mogą działać jako:

- systemy otwarte
- systemy sprzężenia zwrotnego decyzji
- systemy sprzężenia zwrotnego informacji

Warianty przesyłu informacji



system otwarty – przesył w jedną stronę, np. satelitarne misje badawcze – tylko kodowanie informacji

system ze sprzężeniem zwrotnym decyzji – odbiorca przesyła recenzję – transmisja poprawna lub błędna (żądanie retransmisji). Najczęściej stosowane rozwiązanie obsługi błędów w transmisji informacji

system z informacyjnym sprzężeniem zwrotnym – gdy odbiorca nie podejmuje decyzji (b. proste urządzenie) lub gdy celem jest dokładna ocena częstości pojawiających się błędów (*BER*) – pomiar identyfikuje także błędy niewykrywalne metody kodowania

Komunikacja urządzeń cyfrowych

Urządzenia cyfrowe w ogólnym przypadku do obsługi wymiany informacji mogą wykorzystywać wiele sygnałów (linii sygnałowych).

Linie te mogą sprawdzać gotowość urządzeń do współpracy, wskazywać urządzenie odbierające, sygnalizować początek przesyłu, przesyłać dane, taktować przesył elementów informacji, potwierdzać poprawny odbiór elementów, całości informacji, sygnalizować zakończenie wymiany.

W połączeniach magistralowych do każdego zadania może być dedykowana jedna lub kilka linii sygnałowych, z wykorzystaniem których opracowany jest algorytm obsługi przesyłu informacji.

Przy połączeniu urządzeń pojedynczą linią transmisyjną wszystkie te zadania trzeba zrealizować przesyłając jedną linią sekwencje informacji zgodnie z przyjętą procedurą.

Tą procedurę obsługi nazywa się **protokołem komunikacyjnym**

Stosowane są dwa typy protokołów komunikacyjnych:

- Protokoły o organizacji znakowej
- Protokoły o organizacji bitowej

Protokoły o organizacji znakowej wykorzystują do przekazu sygnałów organizujących wymianę informacji wybrane znaki sterujące, nazywane znakami sterowania transmisją.

Standardowe kody EBCDIC (Extended Binary Coded Decimal Interchange Code) czy ASCII część przestrzeni kodowej rezerwują na różne operacje sterujące wydrukiem, formatowaniem i także transmisją.

W podstawowej (7bit) tablicy kodu ASCII do sterowania transmisją przeznaczone jest 10 kodów z 0 i 1 kolumny tablicy.

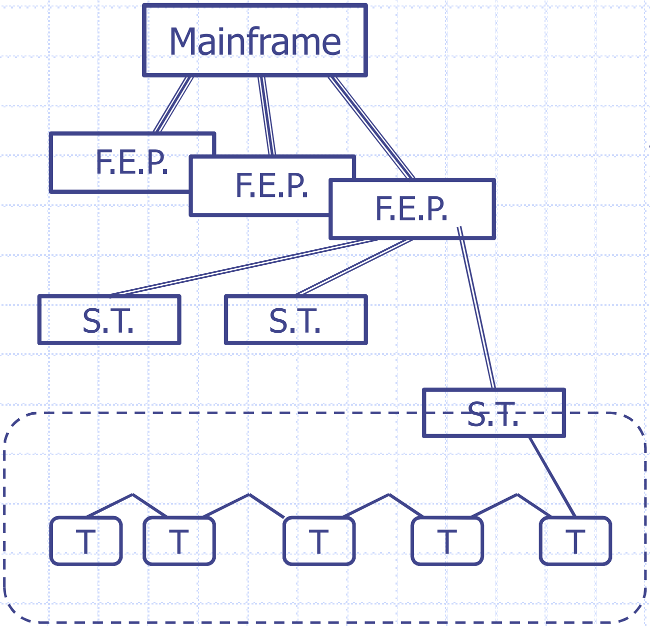
Wykorzystywane są jako separatory części bloku danych, do sterowania przebiegiem transmisji oraz przekazywania potwierdzeń.

USASCII code chart

Bits b_7				b_6				b_5	0 0 0	0 0 1	0 1 0	0 1 1	1 0 0	1 0 1	1 1 0	1 1 1
b_4	b_3	b_2	b_1	Column → / Row ↓	0	1	2	3	4	5	6	7
0	0	0	0	0	NUL	DLE	SP	0	@	P	`	p
0	0	0	1	1	SOH	DC1	!	1	A	Q	a	q
0	0	1	0	2	STX	DC2	"	2	B	R	b	r
0	0	1	1	3	ETX	DC3	#	3	C	S	c	s
0	1	0	0	4	EOT	DC4	$	4	D	T	d	t
0	1	0	1	5	ENQ	NAK	%	5	E	U	e	u
0	1	1	0	6	ACK	SYN	&	6	F	V	f	v
0	1	1	1	7	BEL	ETB	'	7	G	W	g	w
1	0	0	0	8	BS	CAN	(	8	H	X	h	x
1	0	0	1	9	HT	EM	)	9	I	Y	i	y
1	0	1	0	10	LF	SUB	*	:	J	Z	j	z
1	0	1	1	11	VT	ESC	+	;	K	[	k	{
1	1	0	0	12	FF	FS	,	<	L	\	l	|
1	1	0	1	13	CR	GS	-	=	M	]	m	}
1	1	1	0	14	SO	RS	.	>	N	^	n	~
1	1	1	1	15	SI	US	/	?	O	_	o	DEL

Protokół BSC (*Binary Synchronous Communications*)

Pojawił się pod koniec lat 60 jako rozwiązanie dla obsługi zbioru prostych terminali dołączonych do systemów mainframe IBM.



Protokół obsługiwał przepytywanie zbioru terminali podłączonych do systemu mainframe przez sterownik terminali ST linia transmisyjną działającą w trybie pół-duplexowym

Ze względu na różnicę możliwości przetwarzania informacji Stacja Nadrzędna (ST) zarządzała działaniem Stacji Podrzednych (T)

Jednostką wymiany danych był blok danych składający się z nagłówka (*header*) i tekstu:

SYN	SYN	SOH	Header	STX	Tekst	ETB	BCC

Wymiana danych odbywają się według jednego z dwóch schematów, nazywanych odpowiednio *żądaniem nadawania* i *żądaniem odbioru*

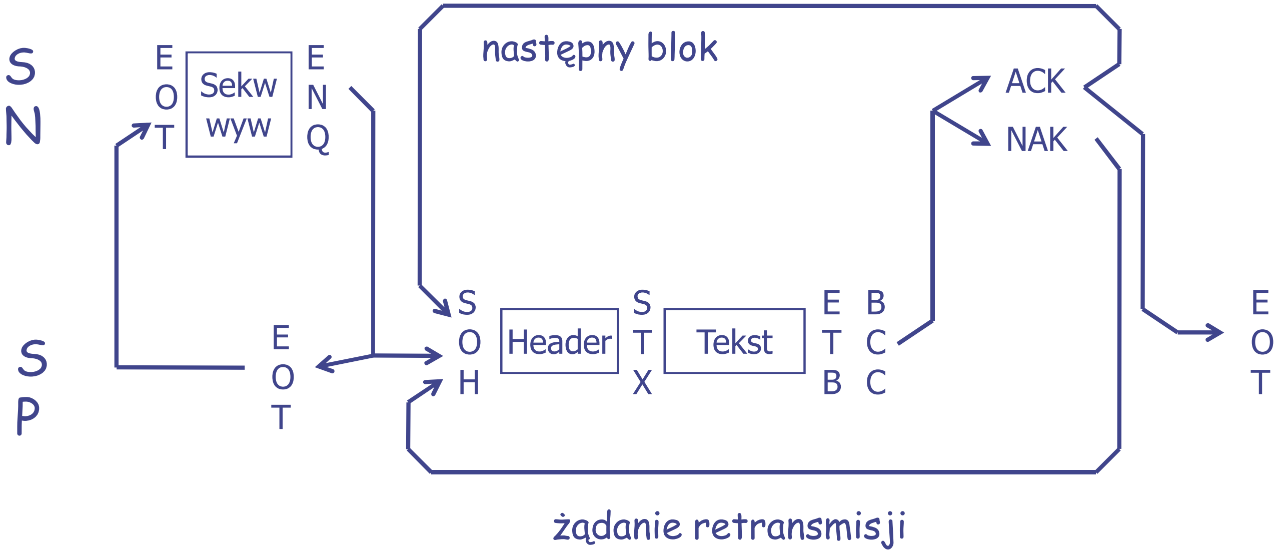
EOT	Sekwencja wywołania	ENQ

Wskazanie schematu transmisji oraz wywoływanej stacji (terminala) odbywa się w sekwencji wywołania wysyłanej przez stacje nadrzędną po zakończeniu poprzedniego żądania

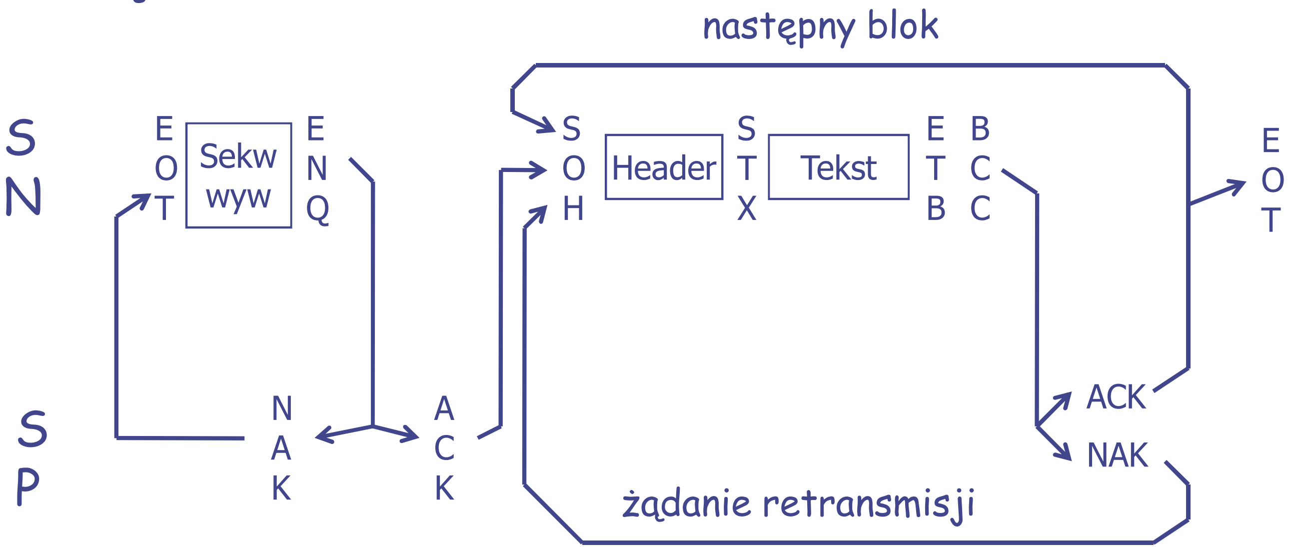
Przesył bloku potwierdzany jest przez odbierającą stację przez wysłanie znaku ACK (potwierdzenie pozytywne) lub NAK (potwierdzenie negatywne).

Po wysłaniu znaków kończących (BCC, ENQ, ACK, NAK) następuje zmiana kierunku transmisji, po której wymagane jest dla synchronizacji zegara wysłanie min. dwóch znaków SYN

Żądanie nadawania



Żądanie odbioru



Inżynieria protokołów komunikacyjnych to zbiór podstawowych rozwiązań, które powtarzają się (są podstawa innych rozwiązań).

Takim rozwiązaniem w BSC jest *obowiązkowe potwierdzenie* (*mandatory confirmation*) – odbiór każdej ramki jest zawsze potwierdzany (ACK lub NAK)

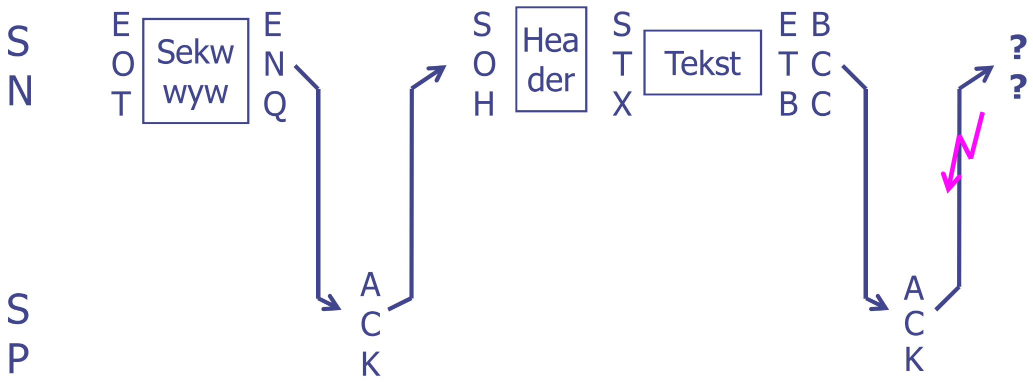
Protokół BSC kontroluje tylko wystąpienie błędu z bloku danych. Sekwencje sterujące, potwierdzenia są przesyłane bez żadnej kontroli.

Podejście takie może wynikać z analizy prawdopodobieństw przekłamania ciągów przesyłanych bitów – jest znacznie większe dla długich sekwencji (blok danych).

Wystąpienie przekłamania np. potwierdzenia może spowodować błędne działanie protokołu, zatrzymanie transmisji (deadlock)

Weźmy np. przebieg żądania odbioru

Stacja nadrzędna wysłała blok, czeka na potwierdzenie

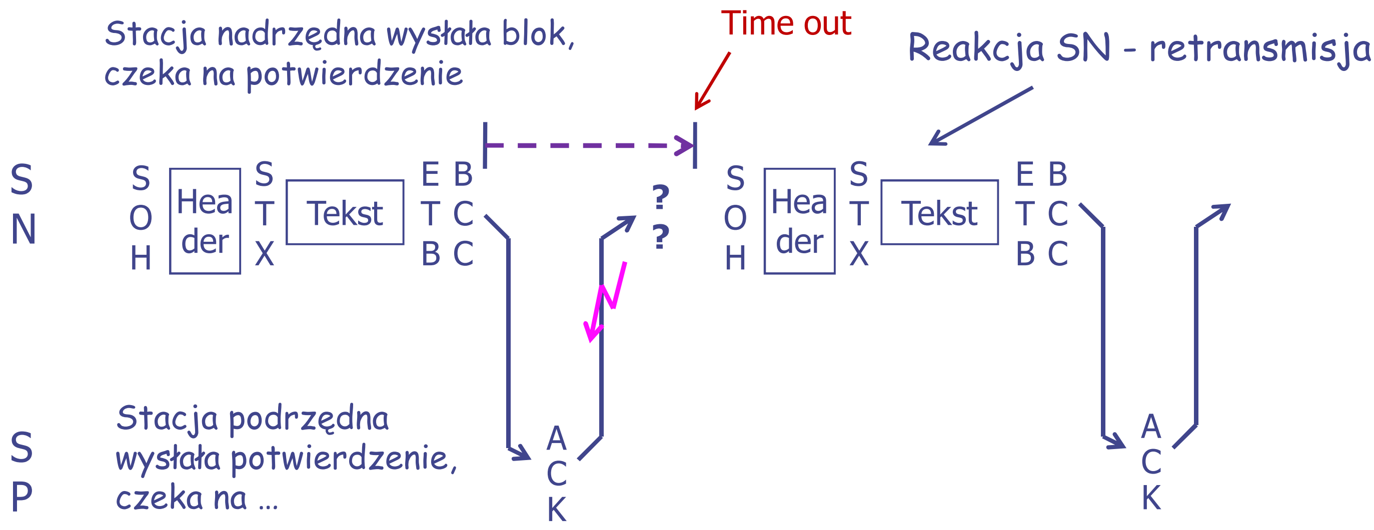


Stacja podrzędna wysłała potwierdzenie, czeka na …

Obie strony wykonały swoje zadanie, czekają na rezultat

Bez interwencji z zewnątrz nic się nie ruszy.

Jedynym rozwiązaniem w tej sytuacji jest wprowadzenie ograniczenia czasu oczekiwania na odpowiedź partnera, przez wprowadzenie *timera* odliczającego zadany czas.
Zakończenie odliczania (*timeout*) może uruchomić specjalną reakcję na zdarzenie.



Czy to skorygowało problem?

Stacja nadrzędna: był błąd, retransmisja, potwierdzony poprawny przesył jednego bloku danych

Stacja podrzędna: odebrany poprawnie, potwierdzony odbiór dwóch bloków danych

To poważny błąd! Jak to skorygować?

Patrząc się na przebieg wydarzeń narzuca się pomysł numerowania przesyłanych bloków danych – stacja podrzędna po numerze łatwo może rozpoznać duplikat.

Problemem jest brak znaków sterujących pozwalających na numerację bloków. Cyfry w kodzie są znakami danych, nie można ich użyć.

Ponieważ w jednym kroku przesyłany jest zawsze jeden blok danych, wymyślono jako rozwiązanie numerowanie potwierdzeń, tzn. w miejsce potwierdzenia ACK wprowadzono dwa potwierdzenia parzyste ACK0 i nieparzyste ACK1.

Jako potwierdzenia numerowane wysyłano sekwencje znaków DLE 0 i DLE 1

Numeracja modulo 2 wystarcza do wykrycia błędu związanego z retransmisją

SN SP

EOT | Sekw. wywołania | ENQ

ACK 0

SOH | Header | STX | Tekst | ETB BCC

ACK 1

Time out, retransmisja

SOH | Header | STX | Tekst | ETB BCC

ACK 0

Problem ! Powinno być potwierdzenie ACK 1

ENQ

Wysłanie znaku ENQ samodzielnie oznacza żądanie resetu transmisji

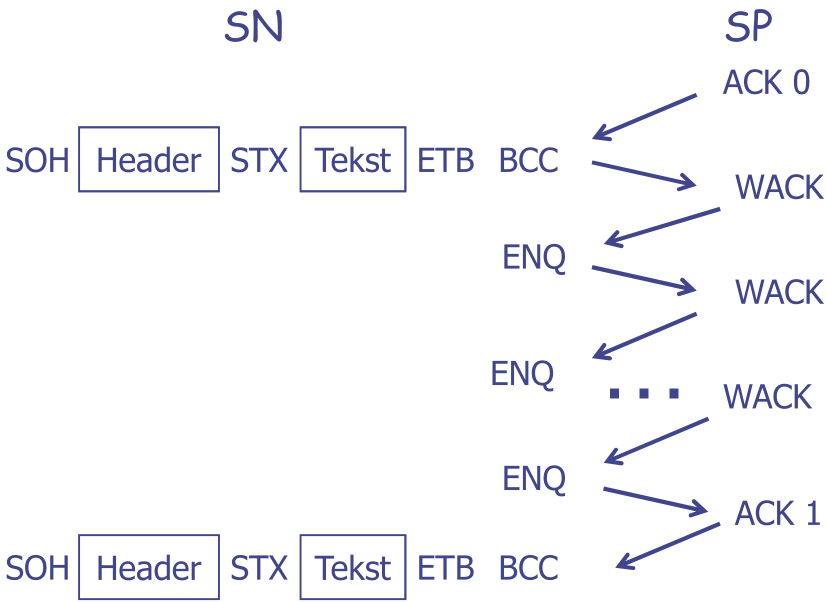
SP kasuje przesłane wcześniej bloki, czeka na sekwencję wywołania

Elementy inżynierii protokołów:

- *Numeracja potwierdzeń* zamiast informacji
- *Time out* jako zabezpieczenie ciągłości działania

Jednym z problemów w komunikacji urządzeń jest przesył między urządzeniami o różnej szybkości odbioru informacji.

Stacja nadrzędna wysyła kolejny blok do urządzenia, które nie zakończyło przetwarzania poprzednio odebranej informacji.



W tej sytuacji NAK, ACK spowoduje kolejną transmisję – potrzebne jest inna odpowiedź, wymijająca –Wait for ACK

Odpowiedź WACK jest wysyłana do momentu zakończenia obsługi bloku

Procedura *sterowania przepływem*

Sterowanie przepływem dopasowuje efektywną szybkość przesyłu do możliwości odbiorcy informacji.

Wymiana *ENQ – WACK* zapewnia ruch danych w łączu, co zapobiega zerwaniu połączenia z powodu nieaktywności łącza

Protokół BSC zaprojektowano do przesyłu danych tekstowych.

Nie można przesyłać nim kodów binarnych programów, symboli graficznych,… – jest *nieprzezroczysty dla informacji.*

Z upływem czasu stało się to wyraźnie niedogodne.

Jak zrobić wersję *przezroczystą dla informacji* kodu BSC, zachowując jego mechanizm działania?

Problemem jest rozpoznawanie i reagowanie na 10 znaków sterujących, mogących wystąpić w przesyłanej informacji.

Rozwiązaniem jest zastąpienie ich jednym.