

# Sieci Komputerowe - ULTIMATE

SonMati

7 stycznia 2015

# Protokół komunikacyjny

Jest to mechanizm, który służy do wymiany informacji pomiędzy dwiema stacjami, jednostkami cyfrowymi (*DD - digital unit*).

## 1 Skład protokołu

- Synchronizacja kooperacji
- Role obu obojga (*parties*)
  - Peer to Peer (P2P)
  - Master-Slave
- *Framing* (Ramkowanie, format przesyłanych danych)
- Poprawa błędów

## 2 Enkapsulacja

### 2.1 5 warstw

Aplikacja
Transport
Sieć
Link access
Fizyczna warstwa

### 2.2 3-poziomowa architektura komunikacji

- File transfer ↔ File transfer (Transport)
- Transport ↔ Transport (Sieć)
- Network Access ↔ Communication network ↔ Network Access (Link access)

## 3 Service Access Point (SAP)

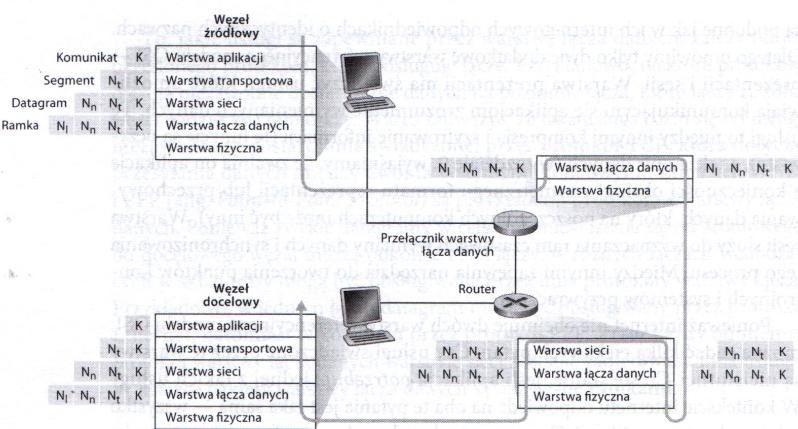
SAP jest wykorzystywany w sieciach typu OSI (*Open Systems Interconnection*), jest to etykietę identyfikującą dla punktów końcowych sieci. Innymi słowy, oznacza ich lokalizację w sieci.

### 3.1 Format

Nagłówek	Host	Punkt przeznaczenia	Dane	Kontrola błędów
----------	------	---------------------	------	-----------------

Kolejność wykonywania wiadomości w protokole: Przykład sieci:

Aplikacja	Dane
Transport	Punkt przeznaczenia + Dane
Sieć	Host + Punkt przeznaczenia + Dane
Link access	Nagłówek + Host + Punkt przeznaczenia + Dane + Kontrola błędów



Rysunek 1.24. Hosty, routery i przełączniki warstwy łącza danych.

Każdy z tych węzłów posiada inny zestaw warstw, które są odzwierciedleniem ich różnego przeznaczenia

## Model ISO / OSI

### 1 7-warstwowy model

Jest to tzw. **stos protokołów**, zaprezentowany od "góry do dołu". Każda warstwa świadczy pewne

Warstwa aplikacyjna
Warstwa prezentacyjna
Warstwa sesji
Warstwa transportowa
Warstwa sieci
Warstwa łącza danych
Warstwa fizyczna

usługi warstwie znajdującej się ponad nią. Innymi słowy, warstwa korzysta z usług tej bezpośrednio poniżej.

- **Warstwa aplikacyjna**

- Zapewnia dostęp do środowiska OSI dla użytkownika
- Udostępnia usługi informacyjne
- Pakiety w tej warstwie nazywamy **komunikatami**

- **Warstwa prezentacyjna**

- Zapewnia niezależność procesom aplikacji od różnic w prezentacji danych (składnia), czyli spełnia rolę tłumacza.
- Jej zadaniem jest świadczenie usług, które umożliwiają komunikującym się aplikacjom zrozumienie wymienianych danych, m.in. kompresja, opis i szyfrowanie danych

- **Warstwa sesji**

- Zapewnia kontrolę struktury dla komunikacji między aplikacjami
- Ustanawia, zarządza i kończy połączenie (sesję) pomiędzy współpracującymi aplikacjami
- Służy do wyznaczania ram czasowych wymiany danych i synchronizowania tego procesu
- Umożliwia tworzenie punktów kontrolnych i systemów przywracania.

- **Warstwa transportowa**

- Zapewnia niezawodny, przejrzysty transfer danych pomiędzy punktami końcowymi
- Zapewnia "od końca do końca" poprawę błędów i sterowanie przepływem.
- Pakiety w tej warstwie nazywamy **segmentami**.
- Przykłady: TCP, UDP

- **Warstwa sieci**

- Odpowiada za ustawnienie, utrzymanie i zakończenie połączenia
- Protokół warstwy transportowej hosta źródłowego przekazuje segment i adres docelowy warstwie sieci
- Umożliwia dostarczenie segmentu do warstwy transportowej docelowego hosta
- Pakiety w tej warstwie nazywamy **datagramami**.

- **Warstwa łącza danych**

- Zapewnia niezawodny transfer informacji przez fizyczne łącze
- Wysyła bloki (*frames*) z niezbędną synchronizacją, kontrolą błędów i kontrolą / sterowaniem przepływem
- Przesyła datagram za pośrednictwem serii routerów
- Pakiety w tej warstwie nazywamy **ramkami**

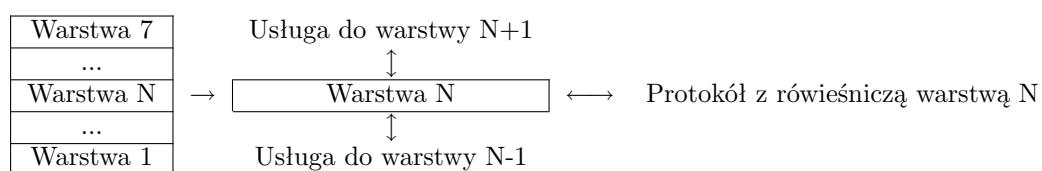
- **Warstwa fizyczna**

- Dotyczy transmisji niestrukturyzowanego strumienia bitów na fizycznym medium
- Dotyczy właściwości mechanicznych, elektrycznych, funkcjonalnych i proceduralnych
- Jej zadaniem jest przesyłanie poszczególnych bitów ramki pomiędzy dwoma węzłami

W zwykłym 5-warstwowym modelu odpowiadające mu warstwy spełniają prawie że te same role.

## 2 Architektura OSI jako Framework

**Wspólny język:** ASN-1 (*Abstract Syntax Notation-One*).  
Model ISO / OSI wykorzystywany jest w technologiach:



- TCP/IP
- LAN
- MAIL
- HTTP

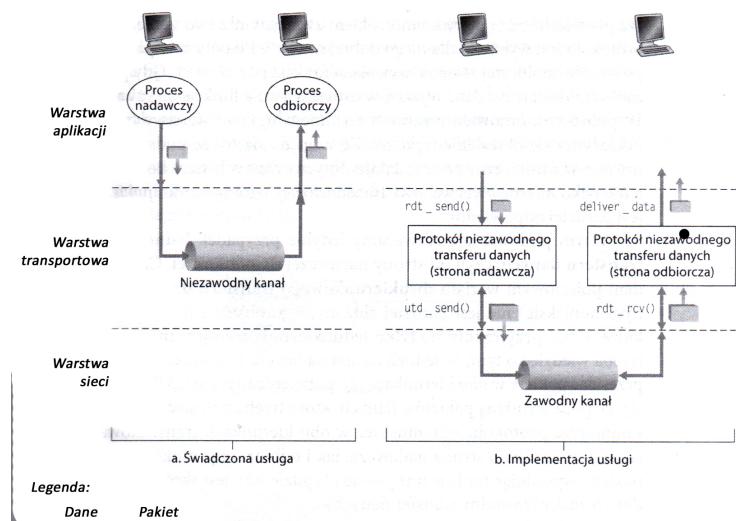
- SECURITY

## Transfer danych

Interesuje nas model **niezawodnego** transferu danych (*reliable data transfer*). Odgrywa on kluczową rolę w pracy sieci, wykorzystywany jest w warstwie transportowej, łącza danych i aplikacji.

Niezawodny kanał zapewnia, że żadne dane nie są uszkadzane, gubione oraz są dostarczane w tej samej kolejności w jakiej zostaną wysłane. Taki transport danych zapewnia np. protokół TCP.

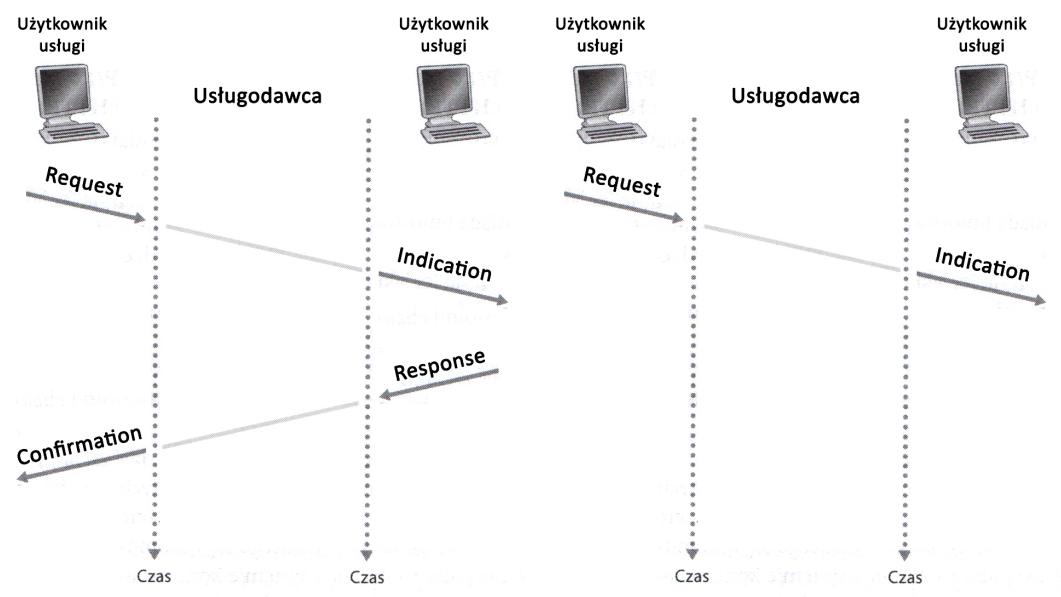
Idea jest prosta, natomiast trudna i złożona jest **implementacja** takiego protokołu.



## 1 Diagramy sekwencji czasowej dla prostych usług

### 1.1 Potwierdzona / niepotwierdzona usługa

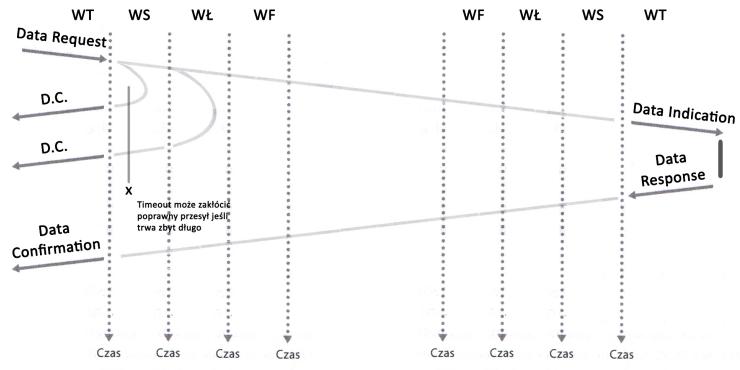
Prosty transport danych z punktu widzenia warstwy aplikacji.



Widzimy tutaj **jednokierunkowy transfer danych** (*half-duplex transfer*) - w danej chwili możliwy jest wyłącznie przesył ze strony nadawczej do odbiorczej.

## 1.2 Wersja z wcześniejszym powrotem

Pełny dwukierunkowy transfer (*Full two-way service*)



Nie jest wiele bardziej złożony od jednokierunkowego

## 2 Protokoły

Wykorzystujemy je do transferu danych pomiędzy warstwami i dzielimy na:

- **Znakowe** - przesyłane są pakiety danych
- **Binarne** - przesyłany jest bit po bicie

## Komunikacja Master-Slave

- 1 A few different algorithms for sending, receiving and processing data
- 2 Master-slave data flow (sliding window)

## Binarna komunikacja synchroniczna (BSC)

Jest to odmiana protokołu znakowego, który służy do implementacji niezawodnego transferu danych z obsługą błędów występujących w kanale zawodnym.  
protokół wykorzystuje mechanizm potwierdzeń

- **Pozytywnych** - poprawna transmisja
- **Negatywnych** - prośba o powtórzenie

Aby móc obsłużyć błędny transfer protokół musi mieć zaimplementowane dodatkowe funkcje:

- **Detekcja błędów** - umożliwienie sprawdzenia u odbiorcy czy wystąpiły w błędne bity.

- **Interakcje ze strony odbiorcy** - w przypadku błędного pakietu odbiorca musi skierować odzew do nadawcy. przykładem są właśnie pozytywne i negatywne potwierdzenia.
- **Retransmisja** - błędny transfer jest ponownie transmitowany.

## 1 Znaki kontrolne

Do realizacji zadań stawianych protokołowi BSC wykorzystano poniższą listę znaków kontrolnych:

- **STX** - Start of Text (początek tekstu)
- **SOH** - Start of Header (początek nagłówka)
- **ENQ** - Enquiry (zapytanie, np. do żądania odpowiedzi)
- **ETX** - End of Text (koniec tekstu)
- **EOT** - End of Transmission (koniec transmisji)
- **ACK** - Acknowledgement (potwierdzenie pozytywne)
- **NAK** - Negative Acknowledgement (potwierdzenie negatywne)]
- **SYN** - Synchronous Idle (czekanie, przed innymi 2 razy)
- **DLE** - Data Link Escape ()
- **ETB**

Najważniejszymi znakami kontrolnymi są ACK oraz NAK - nasze algorytmy muszą tak kontrolować te komendy, by przesył był poprawny i jak najbardziej efektywny.

## 2 Format danych

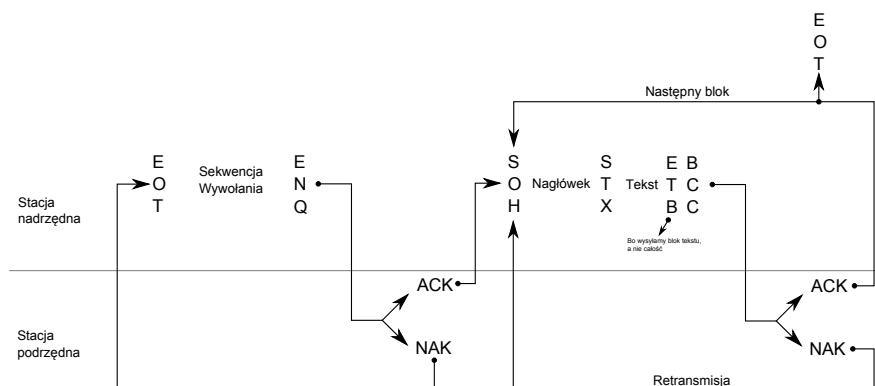
SYN	SYN	SOH	Nagłówek	STX	Tekst	ETX	BCC
-----	-----	-----	----------	-----	-------	-----	-----

## 3 Protokół komunikacji

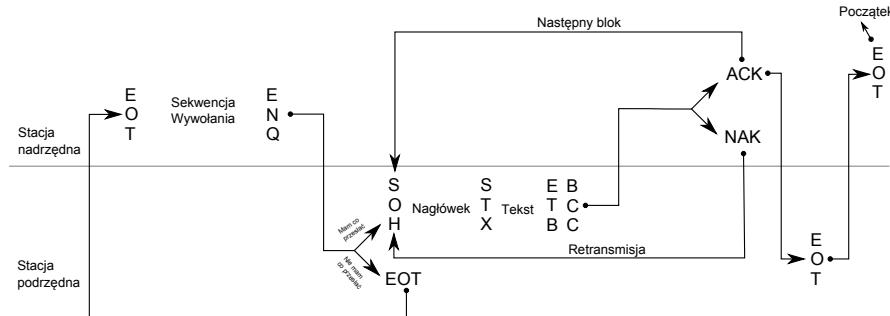
Protokół BSC oparty jest o definicje automatów stanów skończonych (*Finite-State Machine* - FSM) dla strony nadawczej i odbiorczej. Warto zauważyć, że dla obu stron istnieją *niezależne* systemy FSM.

## 4 Schematy żądań

### 4.1 Odbioru



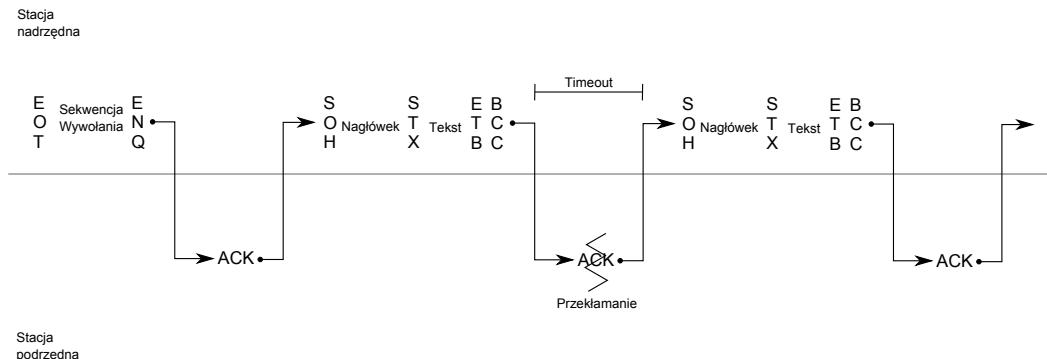
## 4.2 Nadania



## 4.3 Charakterystyka powyższych rozwiązań

- SYN SYN dajemy gdy następuje zmiana kierunku transmisji (SOH, EOT, ACK, NAK)
- Mała kontrola poprawności. Tylko ramka (SOH ... BCC) jest jakoś chroniona. Pozostałe pakiety mogą zostać przekłamane i nadawca w żaden sposób nie dowie się czy odbiorca otrzymał dobre dane. Zatem dodajemy licznik czasu (timeout), dzięki któremu może wykryć błędy jak brak potwierdzenia odbioru.

## 4.4 Z timeout

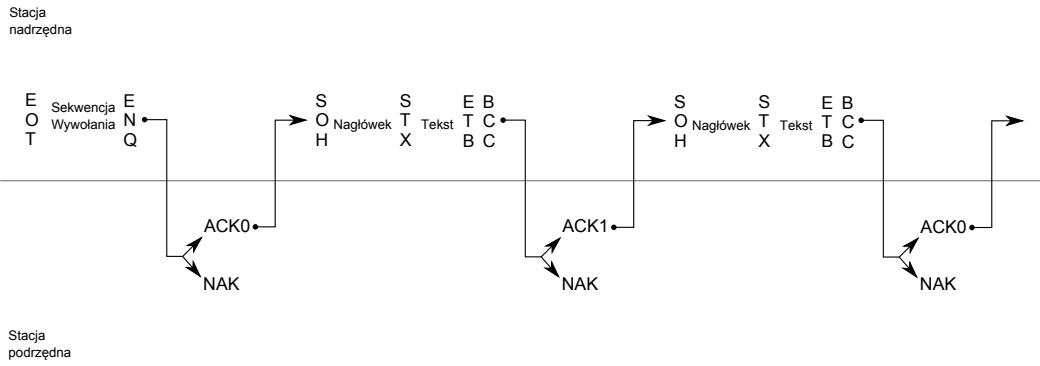


### 4.4.1 Wada rozwiązania

Stacja podrzędna odbiera 2 bloki tekstu, w tym jeden zduplikowany, ponieważ nie została poinformowana przez timeout o przeklamaniu. Odbiorca nie jest w stanie stwierdzić, czy pakiet jest duplikatem czy nową informacją.

## 4.5 Numeracja potwierdzeń

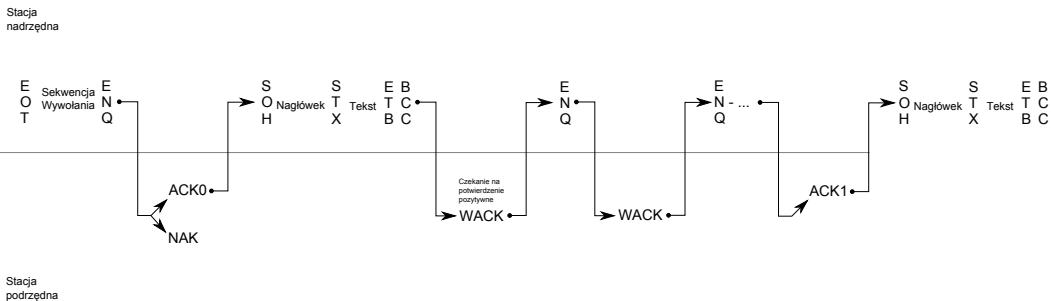
Rozwiązanie problemu duplikatów: wprowadzenie numeru sekwencyjnego. 1-bit, zwiększający się zgodnie z arytmetyką modulo 2, umożliwia stwierdzenie czy pakiet jest wysyłany ponownie lub nie. Może być dodany do ACK lub NAK.



#### 4.5.1 Wada rozwiązańia

Jeżeli stacja nadzędna jest zajęta, to stacja podrzędna tego nie wie. To źle, bo stacja podrzędna zawsze wysyła jakiś sygnał (ACK / NAK).

### 4.6 Sterowanie przepływem



**WACK** - sygnał "czekaj na potwierdzenie". Używany gdy stacja podrzędna potrzebuje czasu na przetworzenie danych.

#### 4.6.1 Wada rozwiązańia

Wadą całego BSC jest brak możliwości korzystanie ze wszystkich znaków przy nagłówku i tekście, bo część z nich jest znakami kontrolnymi. Powstaje limit kombinacji bitów - brak przezroczystości. Skutkuje to tym, że nie można np. wysłać grafiki.

## 5 Przejrzysty blok w BSC



Ten umożliwia przesyłanie grafiki.

Jeśli w nagłówku lub tekście znajduje się znak DLE należy do powtórzyć.

## 6 Przebiegi czasowe

- SN: sekwencja wywołania
- SP: Np. SYN SYN
- SN: Blok
- SP: 4B (SYN SYN DLE0) - czas marnowany
- SN: znów coś

# Protokół o organizacji bitowej SDLC

Protokół SDLC ma za zadanie rozwiązać problem wydajnościowy związanny z BSC. Protokół BSC, który zdefiniowaliśmy pod sam koniec, jest protokołem zatrzymania i czekania. Sprawia, że istnieje zagrożenie nieefektywnego wykorzystania łącza - nadawca może wysyłać znacznie mniej danych w jednostce czasu niż jest to możliwe (nawet tylko ułamek procenta).

Rozwiązaniem jest zastosowanie mechanizmu **potokowego** - polega on na wysyłaniu wielu pakietów bez oczekiwania na potwierdzenie. Np. wysyłając 3 pakiety przed uzyskaniem potwierdzenia może przyspieszyć transfer 3-krotnie.

Zastosowanie tego rozwiązania niesie ze sobą konsekwencje:

- Zwiększenie zakresu numerów sekwencyjnych (każdy musi mieć swój unikatowy numer, a wysyłamy ich więcej)
- Strona nadawcza i odbiorcza protokołu może być zmuszona do buforowania więcej niż jednego pakietu.
- Nowe metody usuwania błędów:
  - Go-Back-N
  - Powtarzanie selektywne

## 1 Łącze, ramki i flagi

### 1.1 Format przesyłanych danych

- **Łączem** przesyłamy ciągły strumień bitów, w nim są separatory.
- Separatory to niepowtarzalne kombinacje, które nazywamy **flagami**.
- Po fladze jest **ramka** i znów flaga. Można powiedzieć, że każda ramka jest otoczona dwiema flagami.

### 1.2 Flaga

Flaga ma wartość 0111110. Pojawia się problem gdy w pakiecie danych jest coś podobnego. Rozwiązanie: algorytm szpikowania zerami. Pomiędzy 5tą i 6tą jedynką wstawia się techniczne zero (011111010). przy odbiorze drugiej flagi się je kasuje.

### 1.3 Format ramki

FLAGA (1B)	Pole adresu (1B)	Pole sterujące (1B)	Dane	CRC (2B)	FLAGA (1B)
------------	------------------	---------------------	------	----------	------------

## 1.4 Adresowanie

Adres zawsze dotyczy terminala (odbiorcy lub nadawcy).  
 ADR = 0xFF —> transmisja rozgłoszeniowa.

### 1.4.1 Pole sterujące

- Typ I (informacyjny)  
 P/F - flaga sterująca kierunkiem transmisji. Jeśli 1 to po zakończeniu odbioru następuje zmiana

1	3	1	3
0	N(S)	P/F	N(R)

kierunku.

- Typ S (sterujący numerowany)

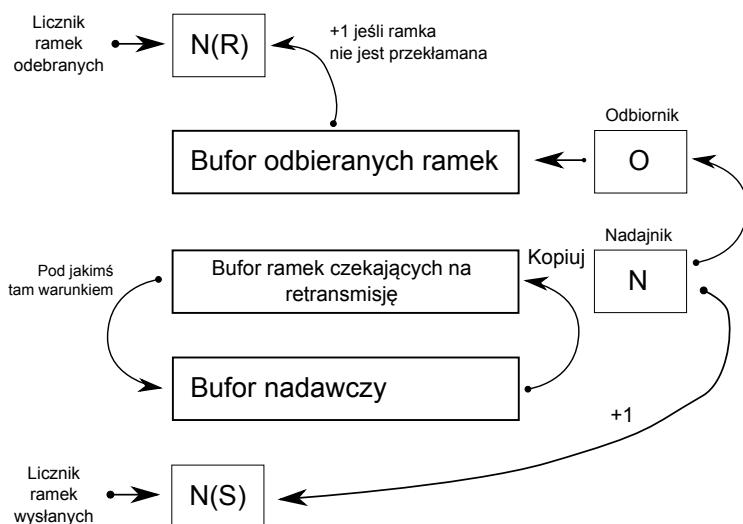
1 0	$S_1 S_0$	P/F	N(R)
-----	-----------	-----	------

- Typ U (sterujący nienumerowany)

W pustych polach można przesyłać polecenia sterujące. Typ wykorzystywany do zamykania i otwierania połączenia oraz do stanów awaryjnych.

- Pierwsza ramka z SN mówi według jakiego algorytmu będzie się odbywał przesył.
- Nowsze tryby przesyłu (z większymi licznikami):
  - NRM - podobne do BSC, wykorzystany typ U. Tryb normalnej odpowiedzi.
  - ARM - obie stacje niezależne, bez P/F. Równoległe asynchroniczne przesyłanie.
  - ABM - przesył równoczesny, usunięcie stacji nadzorujących.
  - W przypadku użycia wersji z 7-bitowym licznikiem istnieją jeszcze 3 tryby: SNRM, SARM, SABM (wszystkie z opcjonalnym E)

## 2 Adapter komunikacyjny



### 3 Komendy sterujące

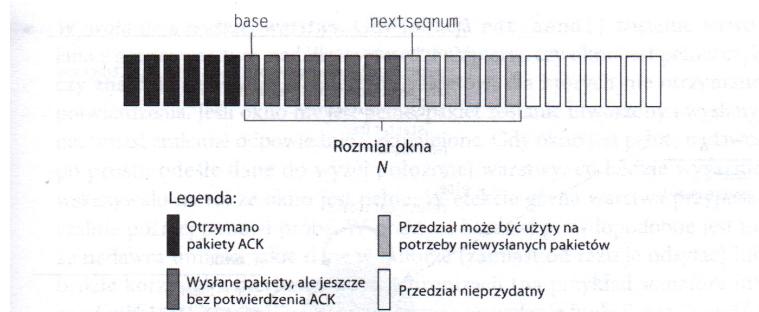
- UA - nienumerowane potwierdzenie
- UI - nienumerowana informacja
- CMDR - odrzucenie polecenia (brak zgodny na dany typ)
- FRMR - odrzucenie ramki niepasującej w ustalony schemat
- RD (*remote disconnect*), DISC, DM (*disconnect mode*) - zamykają połączenie
- XID
- Komendy numerowane
  - RR N(R) - *ready*
  - RNR N(R) - *not ready*
  - REJ N(R) - *rejection*  
Np. REJ, 1 - żądanie retransmisji ramek **od numeru 1**
  - SREJ N(R) - *selective rejection* - żądanie retransmisji wskazanej ramki. Break retransmisji grupowej.  
Np. SREJ, 1 - żądanie retransmisji **tylko** ramki nr 1.

### 4 Połączenia w SDLC

#### 4.1 Normalny tryb odpowiedzi

Wykorzystuje algorytm **Go-Back-N** - "cofnij się o  $N$ ". Polega na prostej zasadzie: protokół może wysyłać wiele pakietów bez oczekiwania na potwierdzenie, jednak liczba niepotwierdzonych pakietów transferowanych w potoku nie może przekroczyć pewnej określonej liczby  $N$ .

Ilustracja problemu:



Rysunek 3.19. Numery sekwencyjne protokołu GBN z punktu widzenia nadawcy

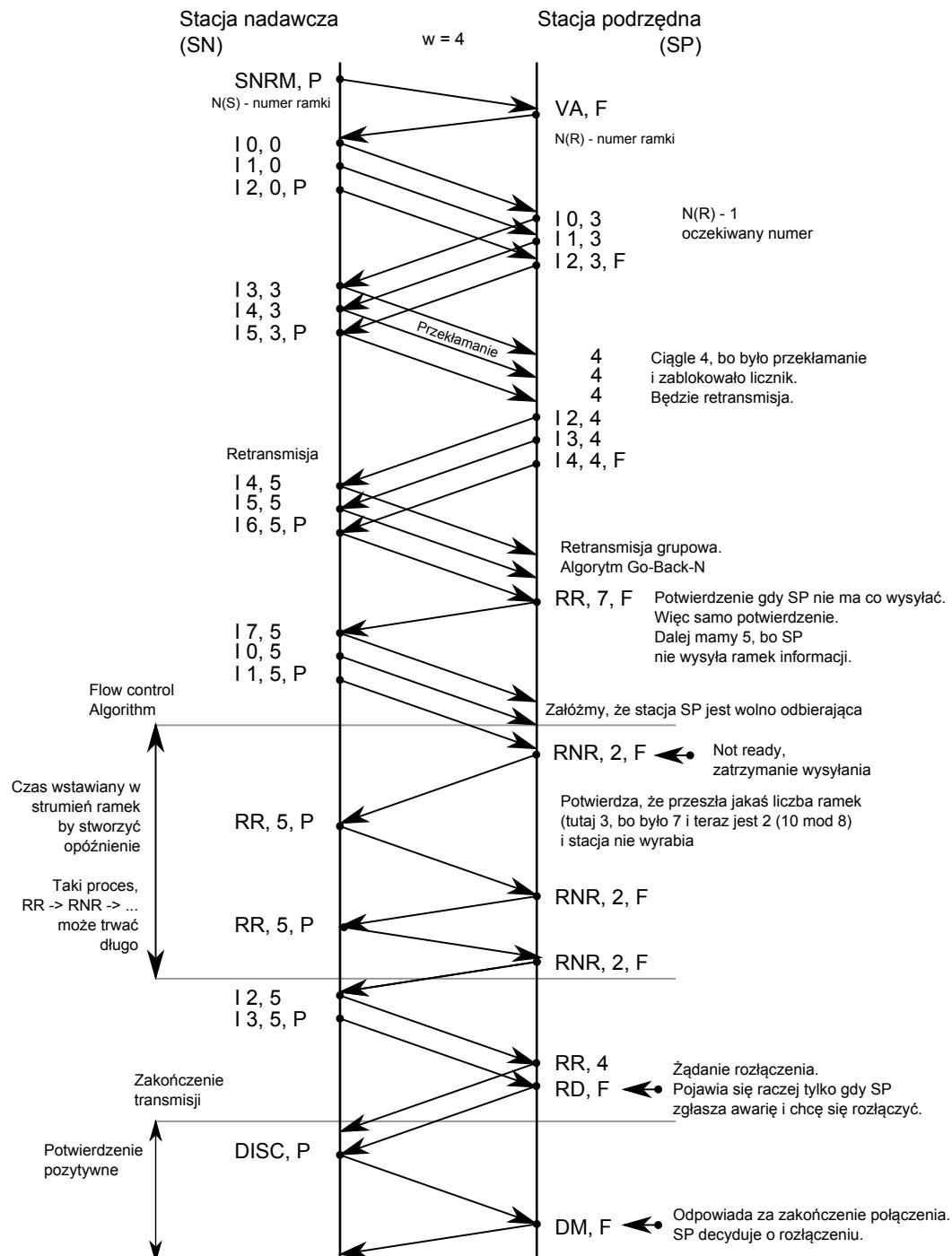
Zakres numerów sekwencyjnych dopuszczalnych w przypadku przesyłanych pakietów, dla których nie otrzymano potwierdzenia, wyróżniono jako **okno** o rozmiarze  $N$ . W trakcie działania okno jest przesuwane wzduż zakresu numerów sekwencyjnych. Dlatego  $N$  nazywamy *rozmiarem okna*, a sam protokół GNB *protokołem przesuwnego okna*.

W protokole występują **zdarzenia**, które należy rozpoznać i obsłużyć:

- **Wywołanie z wyższej warstwy** - gdy z niej otrzymujemy pakiet, nadawca sprawdza czy okno jest pełne
  - Jeśli nie, pakiet zostaje utworzony i wysłany, a zmienne odpowiednio uaktualnione.
  - Jeśli tak, nadawca odsyła dane do warstwy wyższej, co oznacza zapelnienie okna, i wysyła dane później.
- **Odebranie potwierdzenia ACK** - powiązane z pakietem posiadającym numer sekwencyjny  $n$  zostanie uwzględnione w **potwierdzeniu skumulowanym**, które wskazuje, że wszystkie pakiety mające numery sekwencyjne aż do numeru  $n$  właściwie zostały poprawnie przekazane odbiorcy.

- **Zdarzenie upłynięcia czasu** - związane z utraconymi lub znacznie opóźnionymi pakietami. Zegar jest używany w celu wyeliminowania utraconych pakietów lub tych, dla których nie otrzymano potwierdzenia. Jeśli upłynie określony czas, nadawca ponownie wyśle *wszystkie* pakiety, dla których nie uzyskano potwierdzenia.

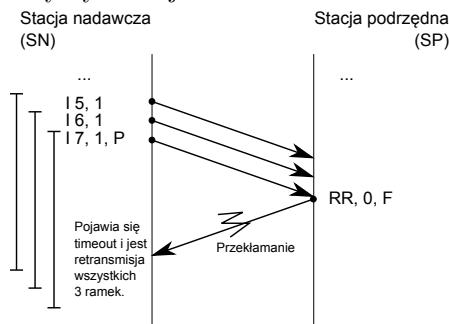
### Przykład:



Optymalną wielkością okna jest 4. Stacja może wysłać tylko 4 ramki naraz, a potem musi czekać na potwierdzenie.

#### 4.1.1 Timeout

Dotyczy każdej ramki



#### 4.1.2 Rodzaje potwierdzeń

- Pozytywne
- Negatywne przez timeout

### 4.2 Tryb asynchronicznej odpowiedzi (ARM)

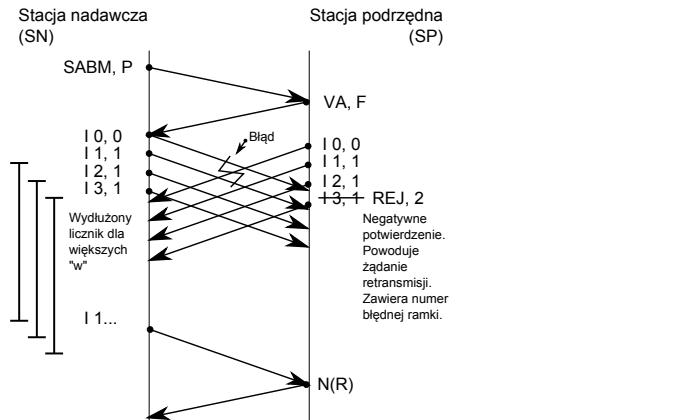
SN i SP mogą wysyłać jednocześnie. Brak sztywnego przypisania na stację podrzędną i nadzorowaną. RNR też tu działa.

#### 4.2.1 Mechanizmy potwierdzenia

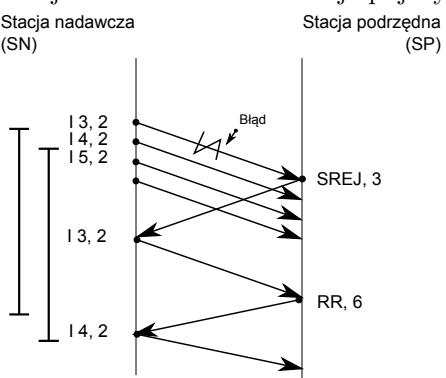
- Pozytywne (stan NCR) - mówi, że nie trzeba się już zajmować którąś z ramek, która została potwierdzona.
- Negatywne (timeout) - wymusza retransmisję.

#### 4.2.2 Właściwości

- Parametr  $w$  pilnuje ilości przesyłanych ramek. Nadmiarowe muszą czekać i są przesyłane później.
- Sekwencja zakończenia podobna do poprzedniego trybu. Może być nawet RD, F, ale musi SP wysyłać potwierdzenia odebrania ramek od SN. Na końcu SN wysyła DISC.
- P i F są potrzebne tylko na początku.
- Timeout powinien być duży, ale nie za duży.



Wersja ze **SREJ** - retransmisja pojedynczej ramki



#### 4.2.3 Zrównoważony tryb działania (AMB (E))

Dotyczy trybu asynchronicznego. E występuje dla dużych liczników. To samo, tylko stacja, która zaczyna nadawać zostaje oznaczona jako "SN".

4.3 Inne

SDLC jest zastrzeżony przez IBM, dlatego powstały rozwiązania alternatywne (HDLC)

## 5 Jakość łącza

Czyli wpływ kontroli błędów na łącze.

### 5.1 Bit Error Rate (BER)

BER =  $10^{-4}$

### 5.1.1 Przykład

Jest do przesłania 2 kB w blokach 256 B,  $w \equiv 3$

$$N = \frac{2[kB]}{256[B]} = 8 \quad (1)$$

Oznacza to, że:

- 1 na 10 tysięcy bitów zostaje przekłamanych
  - 1 na 1250 bajtów zostaje przekłamanych
  - 1 ramka 262 bajtów (6B nagłówka + 256B danych)
  - $\frac{1250}{262} = 3,45$  ramki
  - Co 3.45 ramki jest przekłamane.

**Rozwiązanie:** więcej ramek przesyłanych (narzut organizacyjny protokołu). Wielkość ramek  $\searrow$  → Narzut  $\searrow$  W  $\searrow$  → Narzut  $\searrow$

## 5.2 Cechy

- Stopa błędów

$$q = \frac{\text{ilosc\_bledow}}{\text{ilosc\_bitow\_transmitowanych}} \quad (2)$$

- Pole danych w SDLC wynosi 128B, ale może też być 64, 256, 1024
- Licząc stopę błędy bierzemy całą ramkę wraz z flagami, więc do tego 128 dodajemy 6 (134)
- Optymalna długość pola danych

$$D_{opt} = \frac{H + C \times T_{out}}{2} \times \left[ \sqrt{1 - \frac{4}{(H + C \times T) \ln(1 - q)}} - 1 \right] \quad (3)$$

Gdzie:

- H - długość nagłówka
- C - przepustowość łącza

## 5.3 Przykład

Dane:

- $C = 9600[\text{bps}]$  (bit na sekundę)
- $T = 50[\text{ms}]$  (timeout)

Dla:

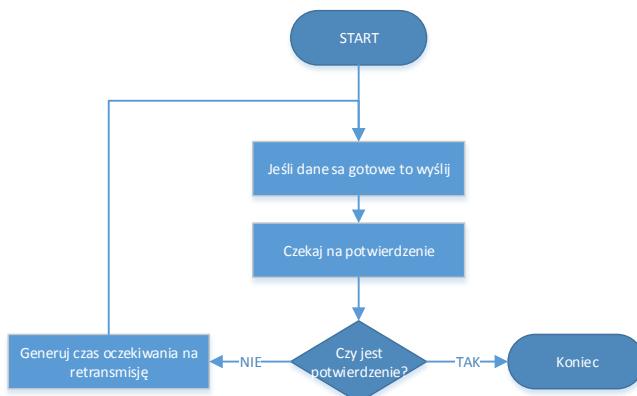
- $q = 10^{-3}$ ,  $D_{opt} = 715[b] = 89[B]$
- $q = 10^{-4}$ ,  $D_{opt} = 2262[b] = 282[B]$
- $q = 10^{-5}$ ,  $D_{opt} = 7155[b] = 854[B]$

**Wniosek:** im mniejsza stopa błędu tym dłuższe optymalne pole danych.

## 6 Łącze radiowe

- Dwa kanały: pierwszy z centrum komputerowego, drugi do CK.
- Problem dostępu do łącza: wbudowanie w terminal algorytmu dostępu do łącza:

### 6.1 Algorytm dostępu do łącza



Retransmisja następuje z losowym opóźnieniem by ramki ponownie się na siebie nie nałożyły.

## 6.2 Protokół

Zastosowano protokół swobodnego dostępu, ALOHA.

# 7 Komunikacja kablowa

## 7.1 Model

Jeden kabel, wiele stacji. Jak chce nadawać to nadaje, tylko czeka na potwierdzenie, jeśli nie nadaje, to dopiero retransmisja (algorytm swobodnego dostępu) (chyba).

## 7.2 Algorytmy

- CSMA (*Carrier Sense Multiple Access*) - wykrywa występowanie zakłóceń, objawiają się w postaci dwóch ramek. Po wykryciu końca kolizji decyduje, która ramka czeka, a która jest retransmitowana.
- p-CSMA (*persistent CSMA*)  
Punkt startu
- Algorytmy, które przerywają transmisję po wykryciu początku zakłócenia danej ramki
  - CSMA / CD - wykrywanie kolizji, początek ETHERNETu
  - CSMA / CA - zapobieganie kolizji, później, wcześniejsze zastosowanie w radiówce
  - CSMA / CR - rozstrzyganie kolizji, lepszy wygrywa i kontynuuje przesył, słabszy musi znów wysłać

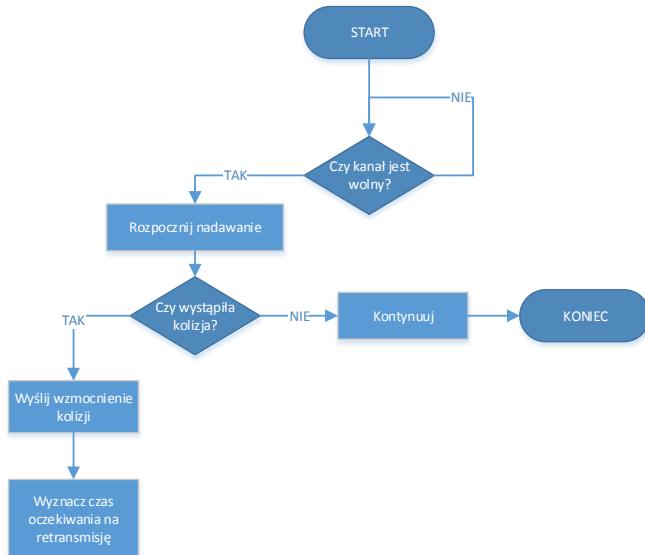
# 8 Dostęp do łączna

## 8.1 Protokoły dostępu do łączna

- Swobodnego dostępu (np. ALOHA)
- Częściowo kontrolowanego dostępu (np. CSMA / ...)
- Kontrolowanego dostępu - faza wykonania bez kolizji i faza gdzie może być.  
Sposoby kontroli:
  - Planowanie transmisji
  - Przekazywanie żetonu (token)

## 9 Częściowo kontrolowany dostęp (działanie CSMA / ...)

### 9.1 Algorytm dostępu



Algorytm oparty o kilka prostych zasad

- "nie przeszukadzaj" (sprawdza czy jest widoczna nośna) - zaczynamy nadawać gdy kanał się zwolni  
CSMA persistent,  $P(\text{startu}) = 1$ . Możliwe kolizje.  
CSMA p-persistent,  $P(\text{startu}) = p$ . (jakoś tak)
- "czy koliduje?" (patrz wyżej)

### 9.2 Czas do retransmisji

Oznaczany jako  $T_R$

$$T_R = r(x) \times 2^k \times T_{ob}$$

Gdzie:

- $r(x)$  - liczba losowa z zakresu 0...1
- $T_{ob}$  - czas obiegu łącza, w najgorszym wypadku podwojony czas propagacji
- $k$  - liczba kolizji, czyli który raz się te ramki już zderzyły. Próbujemy szczęścia aż do  $k \leq 16$

### 9.3 Cechy

Maksymalnie 2800 m do pokonania.

Najgorszy czas  $47,2 \mu s$ .

$51,2 \mu s = 64B$  wyśle. W tym czasie mogą się zdarzyć kolizje.

### 9.4 Poprawna transmisja

Ramka musi być dłuższa niż 64B. Taki wymóg ma Ethernet.

#### **9.4.1 Opcje**

- 3 segmenty i 2 repeatery
- 3 segmenty i 4 pół-repeater

## **Sieć typu Ethernet**

### **1 Ramka**

### **2 Zapobieganie kolizji**

#### **2.1 Klasyczny algorytm**

#### **2.2 CSMA/CA**

#### **2.3 CSMA/CR**

#### **2.4 Algorytmy kontrolowanego dostępu**

##### **2.4.1 Z fazą planowania**

Algorytm rezerwacyjny

##### **2.4.2 Z przekazywaniem uprawnienia (Token)**

- Token
- Token Ring
- Kodowanie informacji
- Ramka
- Timery

- 2.5 Sieć pierścieniowa
- 2.6 Cambridge Ring
- 2.7 Algorytm włączania rejestru
- 2.8 Protokół drzewa rozpinanego

## Control Token Communication

- 1 Two types of networks
- 2 Token passing
- 3 Control tokens
- 4 Conditions

## Projektowanie topologii sieci

- 1 Uzgodnienie gotowości udziału w transmisji
  - 1.1 Tryb połączeniowy
  - 1.2 Tryb bezpołączeniowy
- 2 Tryb pracy sieci
  - 2.1 Komutacja kanałów
  - 2.2 Komutacja informacji
- 3 Adresacja
  - 3.1 IP
- 4 Wybór drogi
  - 4.1 Algorytmy stałego wyboru
  - 4.2 Tablica stałego wyboru
    - 4.2.1 Algorytmy adaptacyjne
      - 1. Zwrotne uczenie
      - 2. Algorytm kooperacji
        - Distance vector

- Link state
3. Tablica routingu

## **Connection-oriented communication**

- 1 Routing**
- 2 Back-learning**

## **Protocols**

- 1 Connection-oriented (X.25 / Frame Relay)**
- 2 Connectionless oriented**