

Sieci Komputerowe - ULTIMATE

SonMati

14 stycznia 2015

Protokół komunikacyjny

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

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)

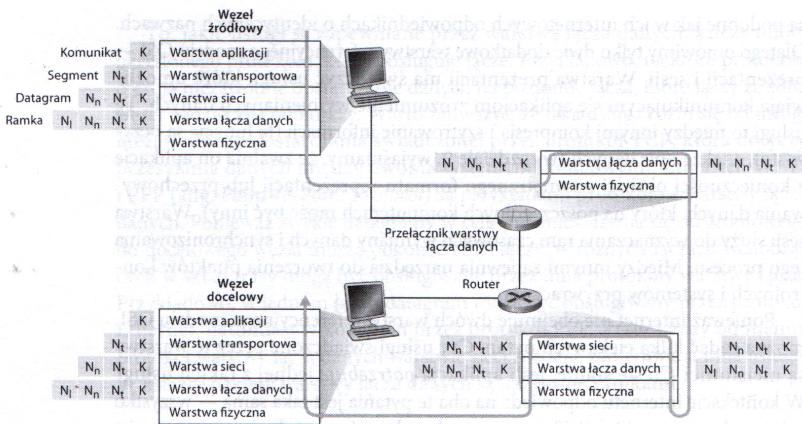
SAP jest wykorzystywany w sieciach typu OSI (*Open Systems Interconnection*), jest to etykietę identyfikującą dla punktów końcowych sieci. Innymi słowy, oznacza ich lokalizację w sieci.

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 ustalenie, 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**

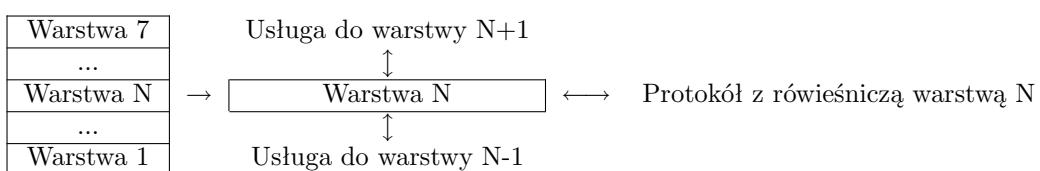
- **Warstwa fizyczna**

- 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

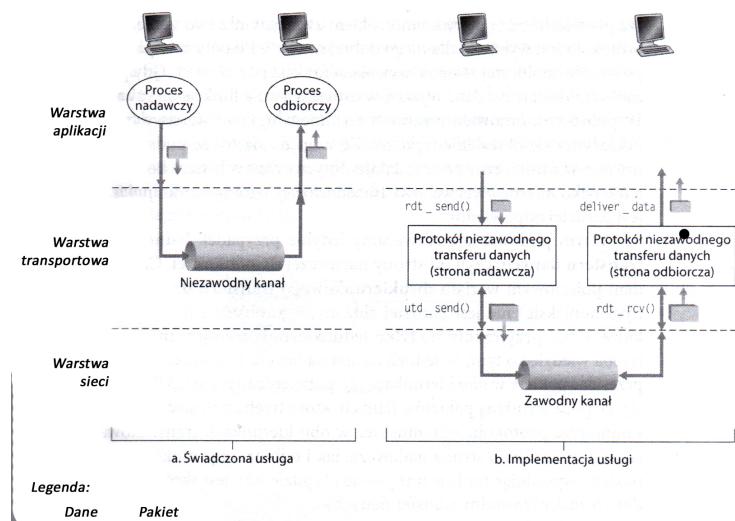
- SECURITY

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.

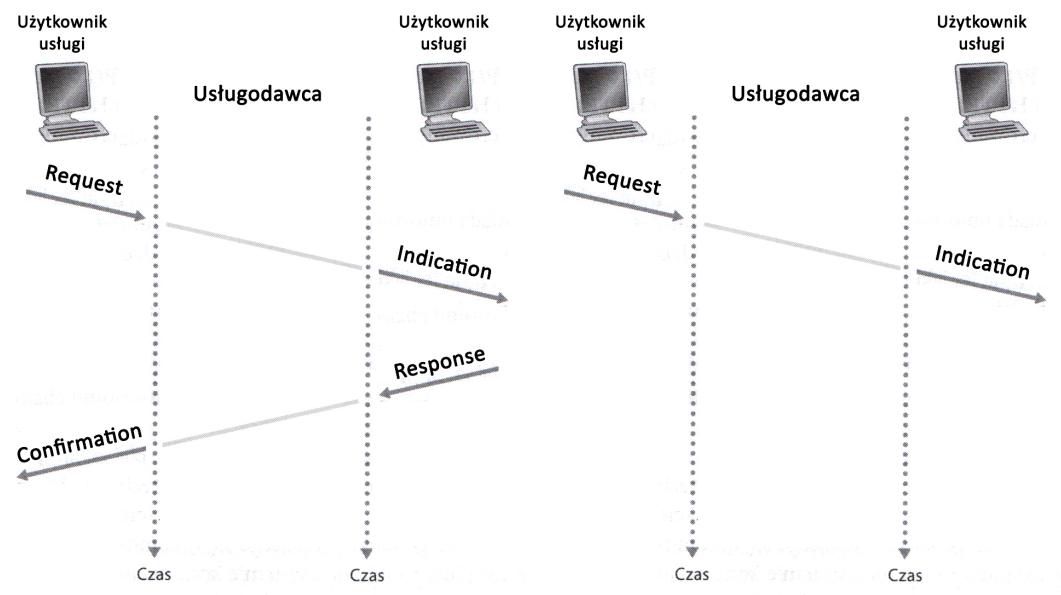
Idea jest prosta, natomiast trudna i złożona jest **implementacja** takiego protokołu.



1 Diagramy sekwencji czasowej dla prostych usług

1.1 Potwierdzona / niepotwierdzona usługa

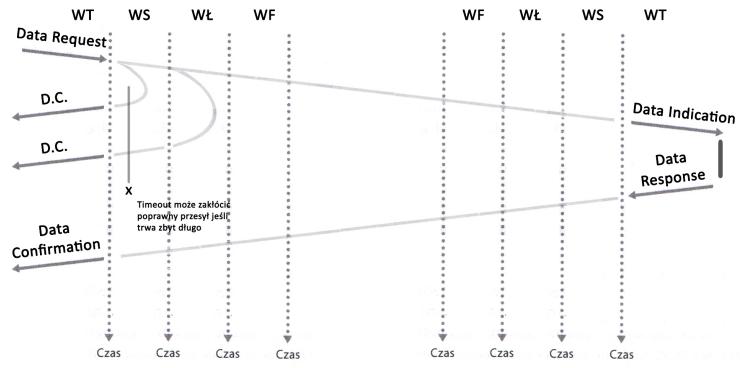
Prosty transport danych z punktu widzenia warstwy aplikacji.



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

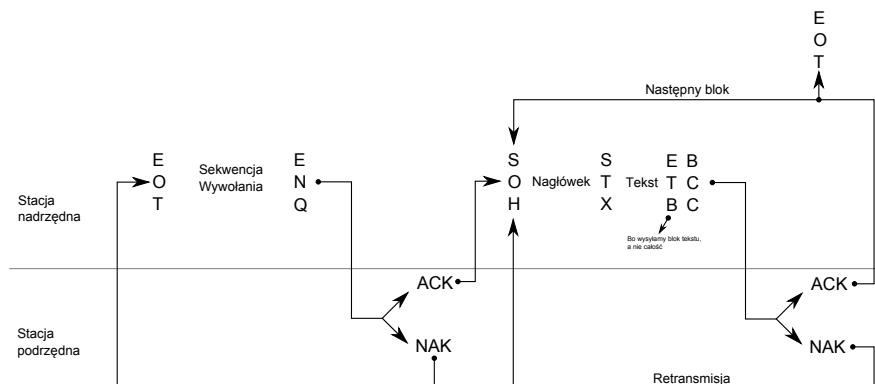
SYN	SYN	SOH	Nagłówek	STX	Tekst	ETX	BCC
-----	-----	-----	----------	-----	-------	-----	-----

3 Protokół komunikacji

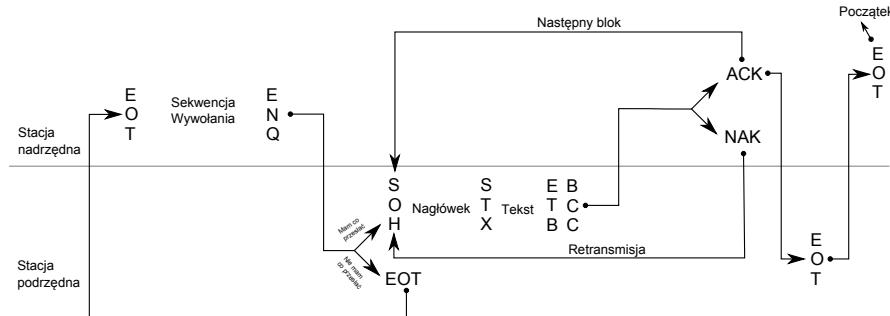
Protokół BSC oparty jest o definicje automatów stanów skończonych (*Finite-State Machine* - FSM) dla strony nadawczej i odbiorczej. Warto zauważyć, że dla obu stron istnieją *niezależne* systemy FSM.

4 Schematy żądań

4.1 Odbioru



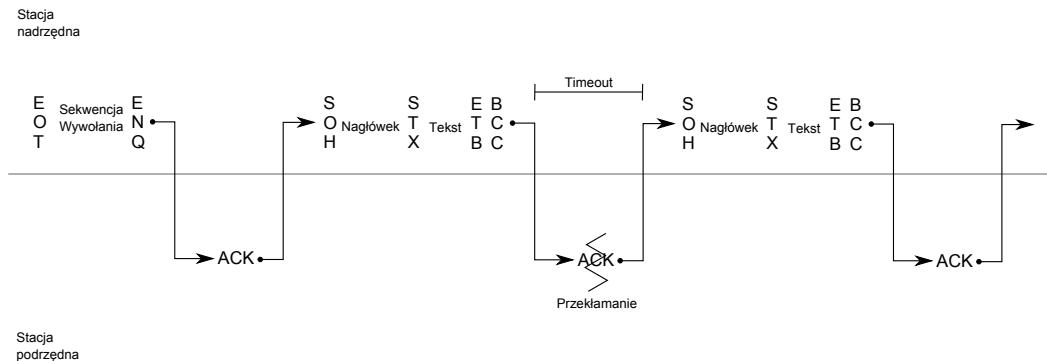
4.2 Nadania



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

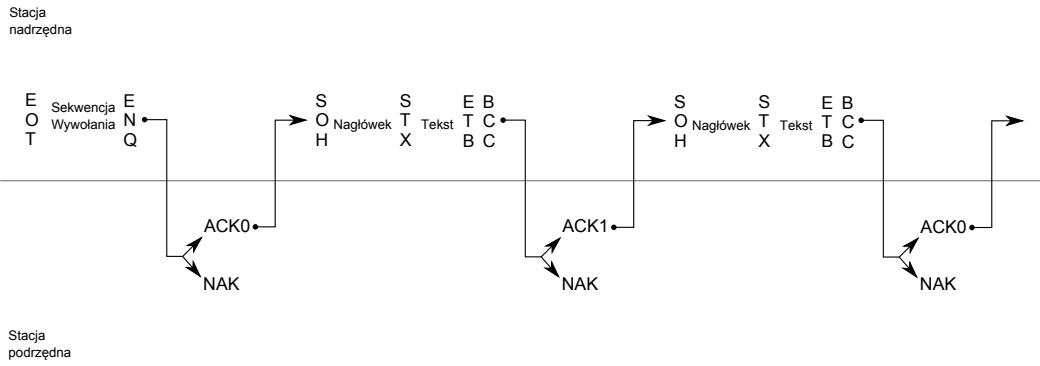


4.4.1 Wada rozwiązania

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

4.5 Numeracja potwierdzeń

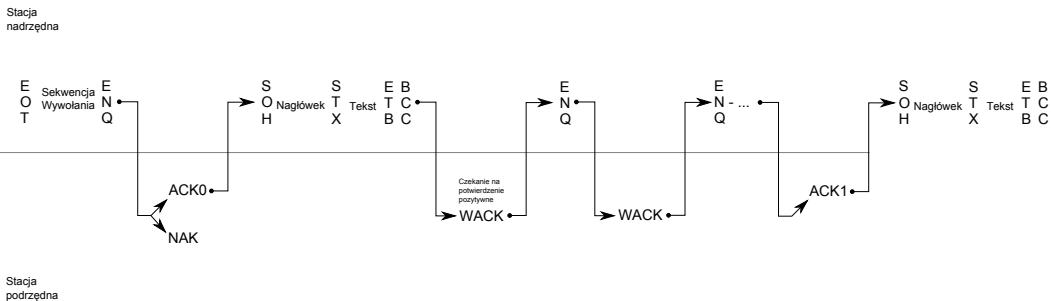
Rozwiązanie problemu duplikatów: wprowadzenie numeru sekwencyjnego. 1-bit, zwiększający się zgodnie z arytmetyką modulo 2, umożliwia stwierdzenie czy pakiet jest wysyłany ponownie lub nie. Może być dodany do ACK lub NAK.



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)

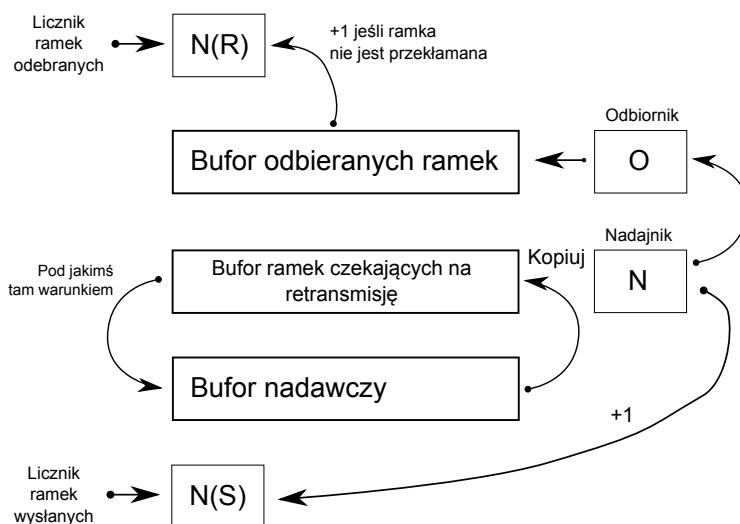
W pustych polach można przesyłać polecenia sterujące. Typ wykorzystywany do zamykania i otwierania połączenia oraz do stanów awaryjnych.

1	1	P/F	
---	---	-----	--

• 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

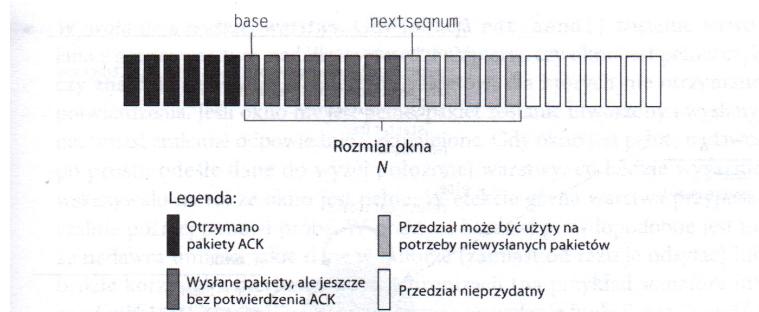
- UA - nienumerowane potwierdzenie
- UI - nienumerowana informacja
- CMDR - odrzucenie polecenia (brak zgodny na dany typ)
- FRMR - odrzucenie ramki niepasującej w ustalony schemat
- RD (*remote disconnect*), DISC, DM (*disconnect mode*) - zamykają połączenie
- XID
- Komendy numerowane
 - RR N(R) - *ready*
 - 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

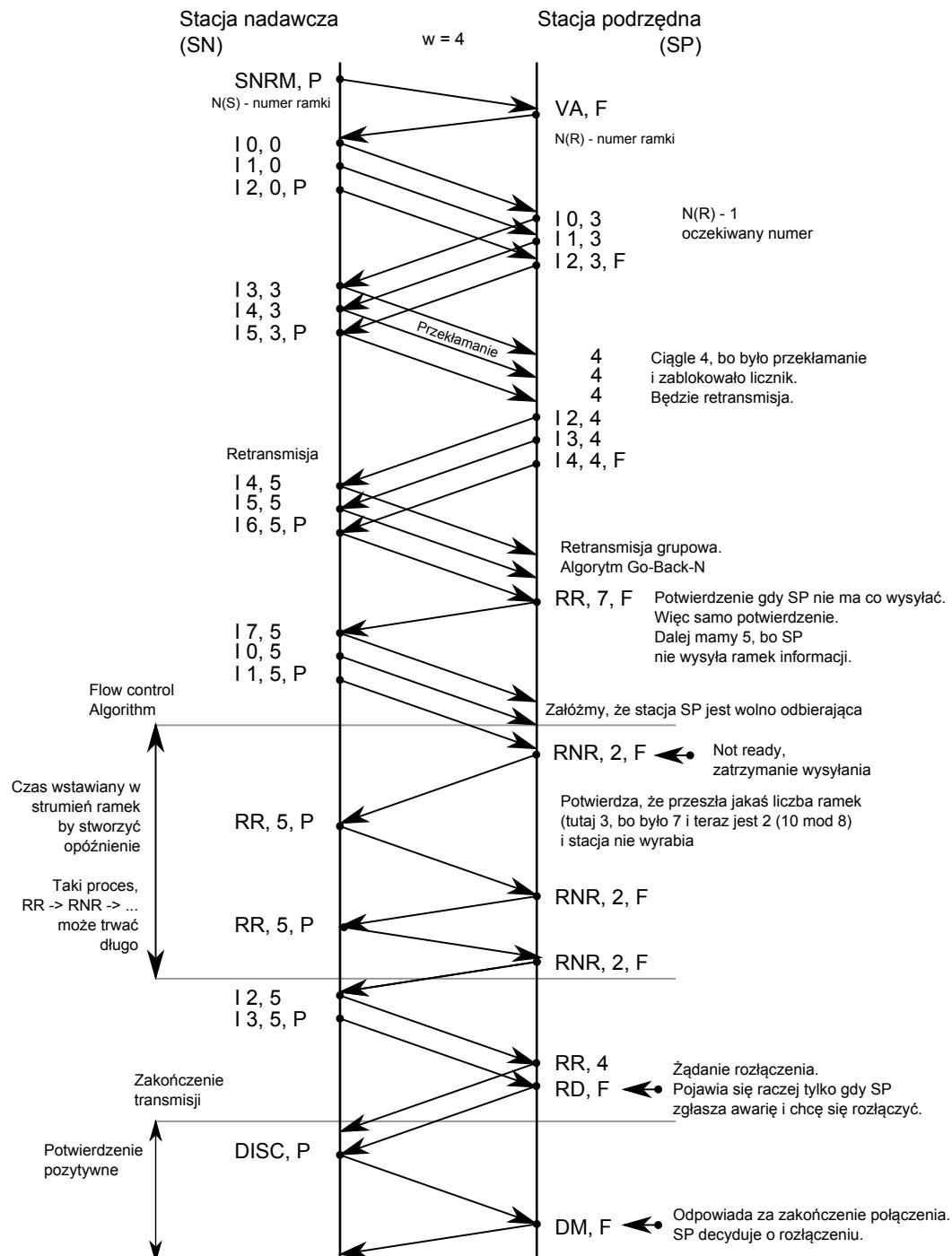
Zakres numerów sekwencyjnych dopuszczalnych w przypadku przesyłanych pakietów, dla których nie otrzymano potwierdzenia, wyróżniono jako **okno** o rozmiarze N . W trakcie działania okno jest przesuwane wzduż zakresu numerów sekwencyjnych. Dlatego N nazywamy *rozmiarem okna*, a sam protokół GNB *protokołem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

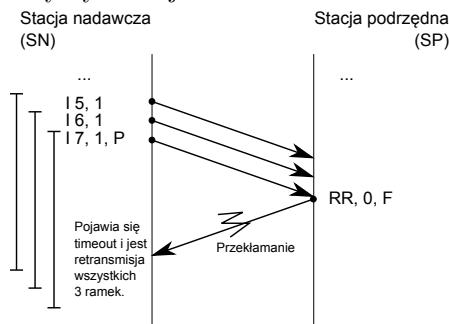
Przykład:



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)

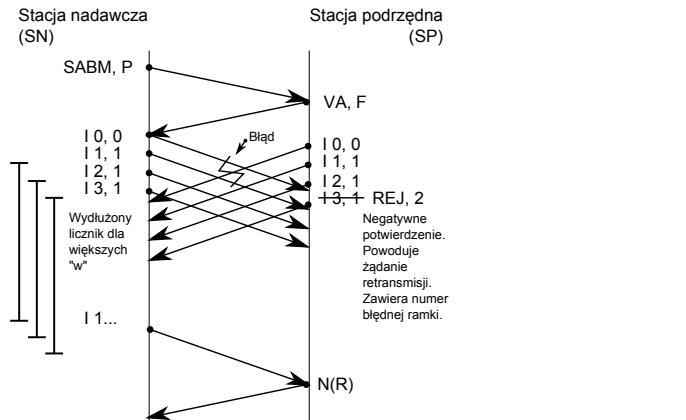
SN i SP mogą wysyłać jednocześnie. Brak sztywnego przypisania na stację podrzędną i nadzorowaną. RNR też tu działa.

4.2.1 Mechanizmy potwierdzenia

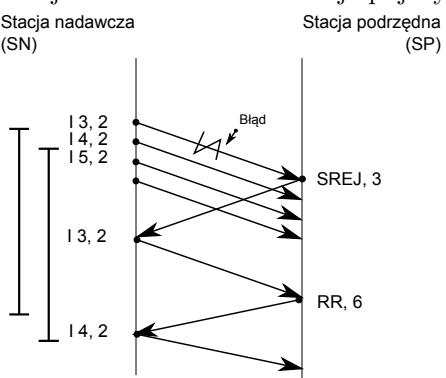
- Pozytywne (stan NCR) - mówi, że nie trzeba się już zajmować którąś z ramek, która została potwierdzona.
- Negatywne (timeout) - wymusza retransmisję.

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 ↘→ Narzut ↘ W ↘→ Narzut ↘

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:

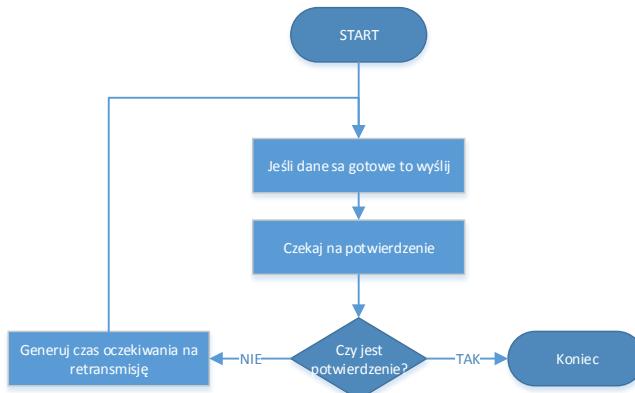
- $q = 10^{-3}$, $D_{opt} = 715[b] = 89[B]$
- $q = 10^{-4}$, $D_{opt} = 2262[b] = 282[B]$
- $q = 10^{-5}$, $D_{opt} = 7155[b] = 854[B]$

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

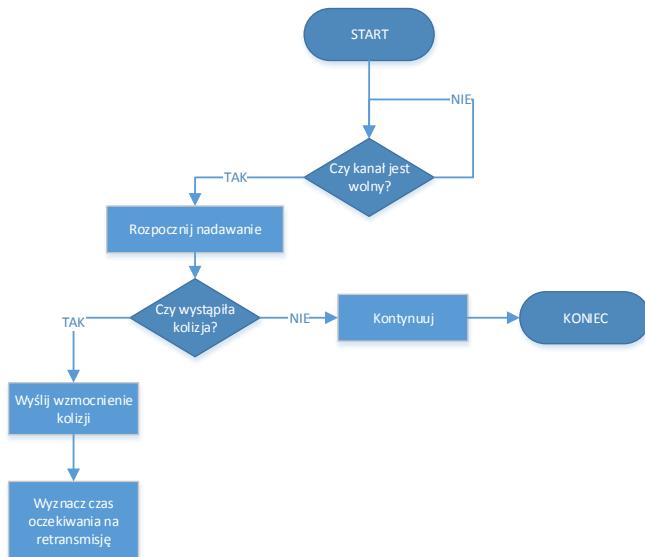
8 Dostęp do łączna

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 przeszukaj" (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 koliduję?" (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

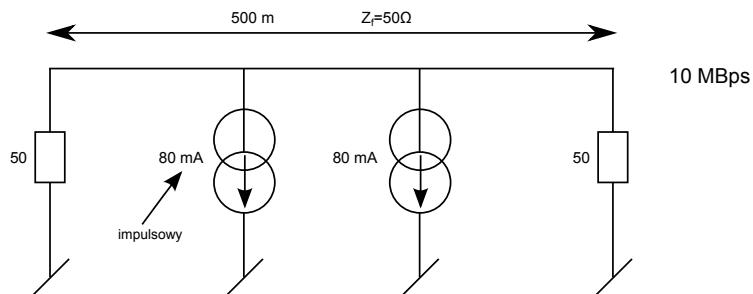
- 3 segmenty i 2 repeatory
- 3 segmenty i 4 pól-repeater

Sieć typu Ethernet

1 Budowa

Standard Ethernet 2.0 wykorzystuje kabel koncentryczny.

1.1 Kabel koncentryczny

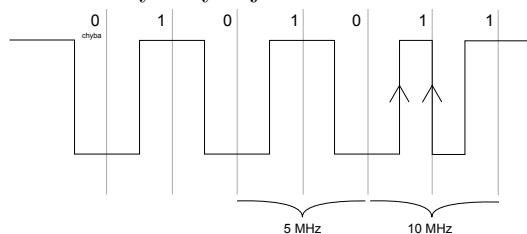


Parametry:

- $U_{IC} = U_p \times e^{\alpha l}$
- $\alpha l \leq 6dB|_{5MHz}$
- $\alpha l \leq 8.5dB|_{10MHz}$

1.2 Kodowanie

Standard wykorzystuje kodowanie Manchester.



1.3 Sprawdzanie kolizji

Chyba chodzi o zapobieganie kolizjom na poziomie sprzętowym.

- Sprawdzenie poziomu
- Pomiar wartości średniej prądu
- Wyłapywanie regularności szpilek
- $T_{ob} = 47.7\mu s$

W trakcie transmisji początkowych 64 bajtów ramki może być złe. Mogą wystąpić 3 segmenty kolizji + $4\frac{R}{2}$ + 2 segmenty łącza.

2 Ramka

To jest (chyba) ramka dla CSMA/CD.

Element	Wielkość	Komentarz
Preambuła	8 B	7×10101010 , a na końcu 10101011.
Adres odbiorcy	6 B	MAC
Adres nadawcy	6 B	MAC
Kontrola	2 B	
Pole danych	od 46 B do 1500 B	
CRC	4 B	
Cisza na łączu między znakami	12 B	$9.6 \mu\text{s}$

Każdy adres w Ethernecie powinien kończyć się zerem - adres stacji - jeden oznacza adres grupowy.

3 Zapobieganie kolizji

3.1 Historyczne metody

- Wprowadzenie Hubów - zmiana transmisji z szeregowej na równoległą.
- Switch: usuwa ramki kolizyjne. Zapamiętuje wysyłane ramki, które może bez końca retransmitować → Znosi odpowiednie rozpiętości sieci → wystarczy więcej switchy.
- Logiczny podział na małe sieci lokalne.
- Jumbo frame - pole danych do 9000 B. Wada: faworyzowanie pewnych stacji.

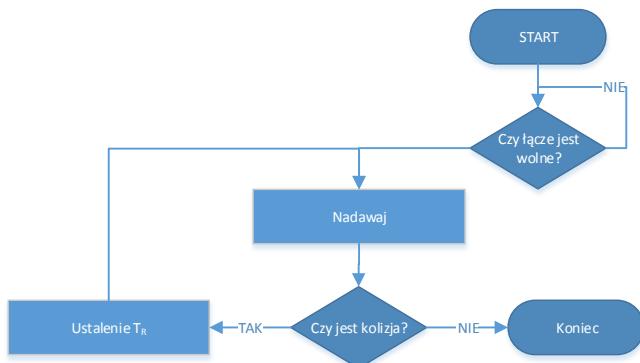
3.2 Klasyczny algorytm

3.2.1 Idea

Kilku chce wysłać, czekaj, retransmitują, znów problem.

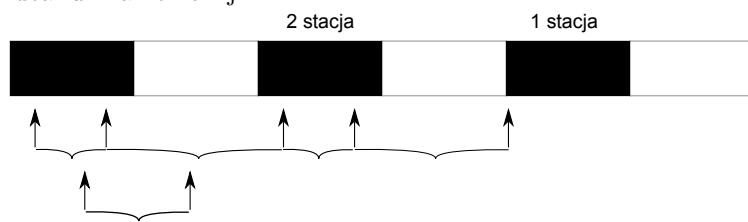
”Nie ma sensu czekać na wolne łącze, lepiej wygenerować pewien czas po którym ponownie sprawdzi się dostępność łącza”.

3.2.2 Schemat blokowy



3.3 CSMA/CA

Idea: unikanie kolizji



Problem: 1 stacja próbuje wysłać, 2 stacja próbuje wysłać.

Analiza: jak widać brak kolizji. Najpierw patrzy czy łącze jest wolne, jeśli nie to generuje czas losowy, do następnej retransmisji. Czyli na początku ta generacja czasu, a nie na końcu jak w klasycznym Etherencie.

3.3.1 Ramka

	2B	2B	6B	6B	6B	2B	6B	0-23(?)B	4B
Preambuła	Frame Control	Duration	A1	A2	A3	Nr sekwencji (?)	A4 (opcjonalnie)	Dane	CRC

3.3.2 Problem

Duże zakłócenia przy dużej ramce.

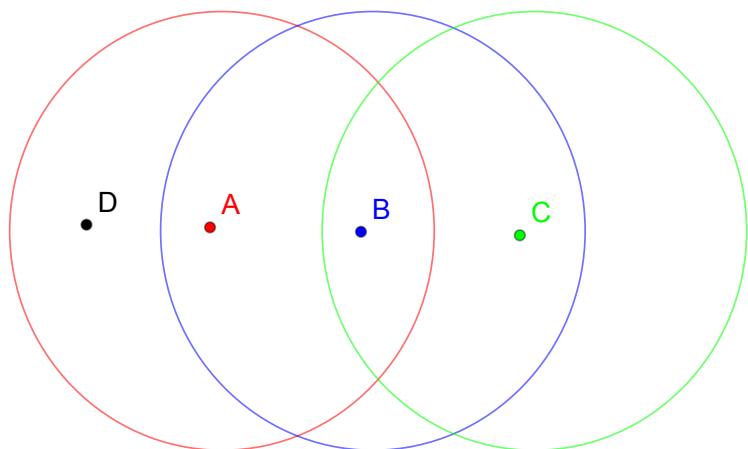
FC i NS wspomagają fragmentację ramki.

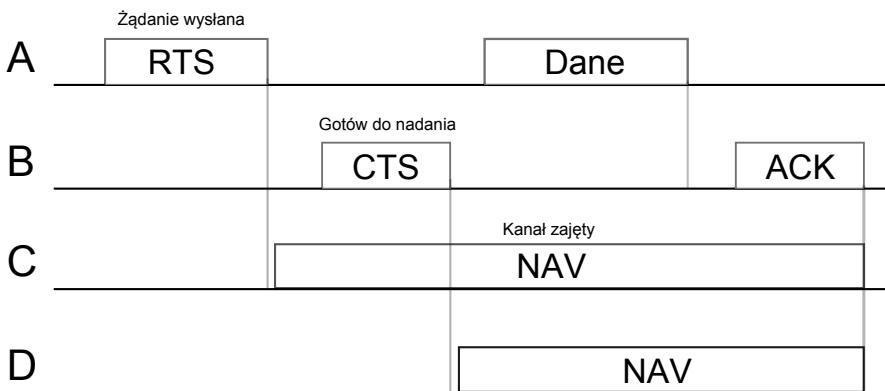
Wraca wzór Shannona z pestek:

$$C = B \times \log_2 \left(1 + \frac{P_s}{P_w} \right)$$

3.3.3 Zmodyfikowane CSMA/CA

Wykorzystywane w radiówce.





Czyli blokujemy pozostałe stacje (C i D), by nie zakłócały.

Stację C blokuje stacja A, a stację D blokuje stację B.

Po stwierdzeniu, że kanał jest wolny, stacja musi poczekać jeszcze, bo może przyjść ACK.

Jeśli stacja ustawi w ramce MORE to NAV u innych stacji jest przedłużony o DURATION + ostatni ACK.

3.4 CSMA/CR

Umożliwia rozstrzyganie kolizji. Wykorzystywane w mikrosieciach (IIC (I^2C)) (HiFi).

Pierwsza część sygnału to pole adresu nadawcy. Najbardziej efektywny. Do sieci 1000 kB/s. S-7 m sieci. Sygnał nadawcy musi się rozejść po całej linii i wrócić, co zabiera ogromną ilość czasu.

3.5 Protokoły MAC (*Media Access Control*)

- Swobodny dostęp (z potwierdzeniem)
- Algorytmy częściowo kontrolowanego dostępu CSMA (bez potwierdzenia)
- Algorytmy kontrolowanego dostępu (z potwierdzeniem / brak potwierdzenia dla transmisji danych)
 - Rezerwacyjne
 - Selekcyjne
 - Jawnego wskazania

3.6 Algorytmy kontrolowanego dostępu

- Algorytm z fazą planowania, gdzie jest dopuszczalne rozstrzyganie kolizji (z limitowaną kolizją).
- Algorytm z przekazywaniem uprawnienia, gdzie jest gwarantowany dostęp do łączna (przekazywanie żetonu (Token))

W tych algorytmach można wyróżnić dwie fazy:

- Planowania - widoczna dla wszystkich stacji, podzielona na szczeliny czasowe.
- Wykonania - superramka, uruchamia (...) szczelin czasowych

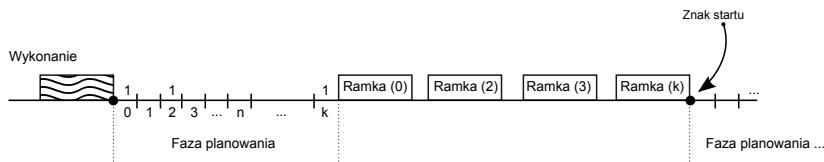
3.7 Algorytmy z fazą planowania

Dopuszczalne rozstrzyganie kolizji (z limitowaną kolizją).

3.7.1 Algorytm rezerwacyjny

Rezerwacja prawa nadawania. Dzieli się na dwie fazy:

1. Rezerwację (faza planowania)
2. Wykonanie (faza wykonania)



Faza planowania - to są szczeliny czasowe.

- Każda stacja ma przypisanie do jednej z nich.
- Jeśli stacja chce transmitować to wstawia 1 w swojej szczelinie czasowej.
- Jeśli nie potrzebuje dostępu, to nie.
- Po 1 bit na każdą szczelinę czasową.

Zwykle grupuje się stacje i każda grupa otrzymuje po jednej szczelinie czasowej (wówczas liczba szczelin < liczba stacji).

Zawsze musi być minimum jedna ramka, bo musi być widoczny koniec transmisji.

Działanie: wysyłamy adres stacji z grupy, ale są kolizje. Tu włącza się metoda generacji czasu na retransmisję itp. Jeśli nie zdąży wysłać, bo już zacznie wysyłać, to czeka na ponowne pojawienie się szczelin czasowych.

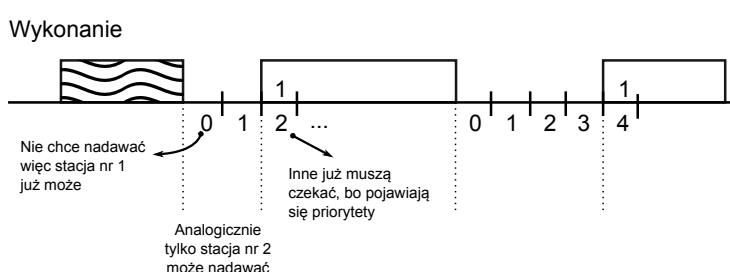
W fazie planowania mogą wystąpić kolizje, ale dla każdej szczeliny kolizje są rozdzielane.

Wykorzystywany w sieciach o dynamicznych priorytetach.

3.7.2 Algorytm selekcyjny

Selekcja wstępna, czyli która stacja ma prawo nadawania. Dzieli się na dwie fazy:

1. Selekcję (faza planowania)
2. Wykonanie (faza wykonania)



Powyższy rysunek przedstawia malenie priorytetu.

Na jedną szczelinę może przypaść więcej niż jedna stacja (grupowanie).

WAŻNE: stacje, które muszą być wykonane są wysyłane jako pierwsze i do tego pojedynczo, a nie w grupie (jedna na jedną szczelinę). Mimo to, zwykle wykonuje się w grupach.

3.8 Algorytmy z przekazywaniem uprawnień (Token)

3.8.1 Co to jest Token?

Token to specjalna ramka sterująca.

Token bus jest zdefiniowany w IEEE.802.4.

3.8.2 Idea działania

Stacja przechowuje adres poprzednika i następcy. Powstaje wówczas lista adresów stacji:
 $A_1 \rightarrow A_2 \rightarrow A_3 \rightarrow \dots \rightarrow A_n \rightarrow \dots \rightarrow A_k \rightarrow A_1 \dots$

Ilustracja przekazywania tokenu.

Parametry:

- Czas korzystania z tokenu: T_k 40kbit
- No Token Timer: $T_{NTokT} = n \times T_k$, gdzie n to liczba stacji (?)

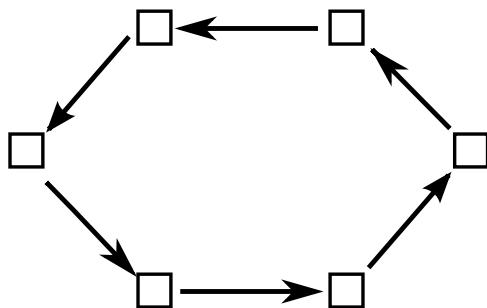
3.8.3 Wada

Wszystko się sypie gdy zostaje przekłamany Token. Stacja czeka, orientuje się, że sieć się sypła, a następnie stacje walczą o to, która stworzy nowy Token.

- Algorytm "głosowania Tokenu", w końcu któraś wygra.
- Algorytm włączania stacji do listy, czyli zgłaszanie następcy (ponowna walka).
- Problem opuszczania sieci.

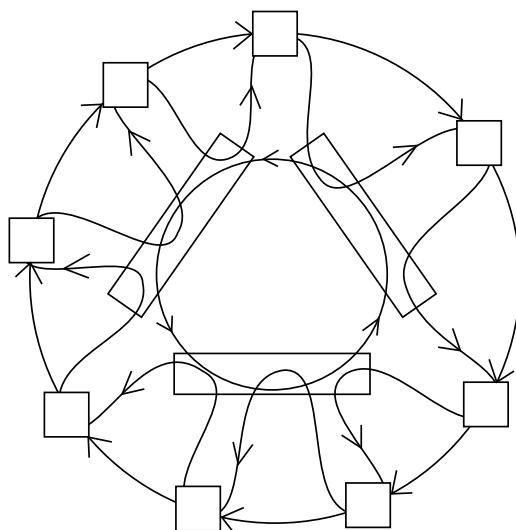
3.9 Sieć pierścieniowa (Token Ring)

Zdefiniowana w IEEE 802.4 (?) Zastosowanie innej (pierścieniowej) topologii sieci. Eliminuje ona wcześniejsze problemy.



- Transmisja tylko w jednym kierunku. Dzięki temu sieć może mieć wiele kilometrów.
- Stacje nie wiedzą co się dzieje u innych.
- Ciężko w niej znaleźć miejsce przerwania.

Sieć dwupierścieniowa (Token Ring).

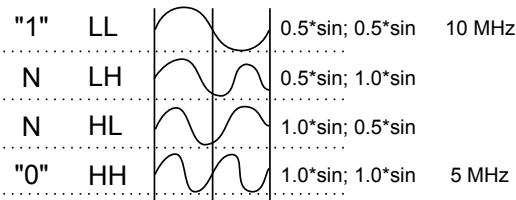


Dwa pierścienie: jeden zewnętrzny, drugi wewnętrzny (rezerwowy). Jeśli jeden przesyła w lewo, to drugi w prawo. Jeżeli zewnętrzny zostanie przerwany, to wewnętrzny zajmuje się załatwianiem tego.

Wada: jeżeli w sieci nie ma Tokenu to występują kolizje.

3.10 Kodowanie informacji

Kodowanie dla sieci opartych o Token.



3.11 Ramka

N	N	0	N	N	0	0	0	SD	start ramki
N	N	1	N	N	1	1	1	ED	koniec ramki

Te "N" mówią, że to SD lub ED, a nie zwykłe dane. Więc tu nie używamy metody szpikowania zerami i tym podobnych.

3.11.1 Token Bus

SD	FC	DA	SA	Dane	CRC	ED
----	----	----	----	------	-----	----

Gdzie:

- SD - Start Delimiter, ma postać NN0NN000, 1B
- FC - Frame Control, czy to transmisja w jednym kierunku czy np. żądanie przesyłu odpowiedzi. Przekazuje informacje sterujące. 1B
- DA - Destination Address (6B / 2B)
- SA - Source Address (6B / 2B)
- ED - End Delimiter, ma postać NN1NN111, 1B

3.11.2 Token Ring

SD	FC	FC	DA	SA	Dane	CRC	ED	FS
----	----	----	----	----	------	-----	----	----

Gdzie:

- AC - adres rozpoznany, 1 - ok, 0 - nie (ramka skopiowana (jakoś tak)). Służy do przekazywania informacji sterujących jak np. wybranie pierwotnego posiadacza tokenu.
- FS - Frame Status

3.12 Timery (dostęp do łącza)

3.12.1 Parametry

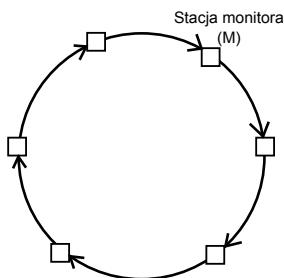
- **THT** (*Token Hold Timer*) - maksymalny czas przetrzymania Tokenu, równy n bitów
- **NTT** (*Not Token Timer*) - czas czekania na token. $NTT = N \times THT$, gdzie N to liczba stacji. Gdy trwa za dugo to sieć pada.
- Problem pojawia się gdy przejdzie cały timer, a tokenu brak.
- Trzeba utworzyć listę adresów log (...). Każda stacja musi pamiętać dwa adresy: poprzednika i następnika.

3.13 Algorytm rozstrzygania kolizji

Dla Token Bus: stacja, która chce coś wysłać, w ramce w FC wysyła CT - zgłoszenie tokenu. Potem stacja walczy z innymi.

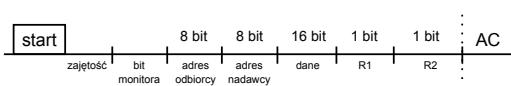
- W sieci pierścieniowej stacje robią opóźnienie 1 bit.
- Stacje walczą o dostęp wg algorytmu rywalizacji. Jeśli adres nadawcy jest niższy niż mój, to ja wysyłam własną. Jeśli jego wyższy to retransmisja (jego ramkę dalej ?).
- Po przejściu całej sieci ramka ma adres zwycięzcy. Przez to opóźnienia. Wtedy ramka dodatkowa, która mówi, że jest już zwycięska i trzeba już normalnie przesyłać.
- W pierścieniowej technologii są potwierdzenia. Na końcu ramki dokleja się informację czy ramka była poprawnie wysłana i odebrana oraz czy zrobiono kopię tej ramki (ta odbierająca robi kopię). Na A lub C musi być jeden, że ok, lub zero, jeśli nie ok. Jeśli A i C = 1, to wszystko ok.
- Czyszczenie sieci pierścieniowej: nadawca coś wysyła, a jeśli coś do niej wchodzi w trakcie to są to śmieci, więc je usuwa. Usuwa też swoją ramkę po przejściu przez sieć.

3.14 Cambridge Ring



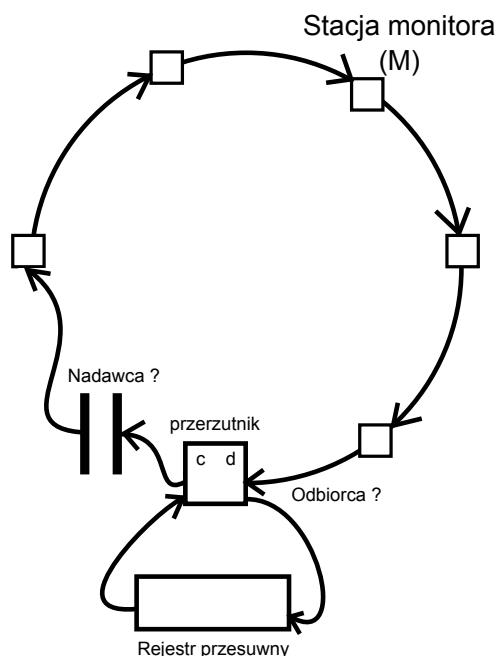
Pojemność informacyjna pierścienia $20\mu s$ gdy $4km$ sieci / 200 bit.
Każda stacja M generuje miniramki o długości 40 bit.

- Liczba bitów w pierścieniu $N_{ring} = \frac{L}{\frac{2}{3}C} \times \frac{1}{V}$, gdzie: L to długość kabla łącza; V to szybkość nadawania.
- Liczba bitów pamiętanych na kablach $n = \frac{T_p}{T_b}$
- $N_{ramek} = N_{ring} + N_{stacji} = 4N/40bit$



- Sieć CR wykorzystuje monitor - wydzieloną stację.
Monitor generuje "strumień" miniramek. $Z = 1$, czyli ktoś zajął tę ramkę i wpisał do niej dane (czyli używa się tej ramki do przesyłu).
 $M = 1$, czyli monitor już tę ramkę widział. Monitor skasuje tę ramkę gdy odczyta ją i ta ramka ma $n = 1$. Za drugim razem ??
Gdy trzy pierwsze bity (startu, zajętości i monitora) są równe 1 to oznacza, że ramka obiegła cały pierścień i monitor wysyła nową tacę (??) informacyjną.
- Szybkość transmisji: $\frac{2B}{T_{ob}}$, gdzie T_{ob} to czas obiegu, który trwa $30 \mu s$.
- Najgorsze rozwiązanie pod kątem serwisowania.

3.14.1 Algorytm włączania rejestru



Dzięki temu może przesyłać duże ramki. Część wchodzi do rejestratora przesuwnego.

3.14.2 Protokół drzewa rozpinanego

Spanning Tree Protocol. Przydatne przy sieci pierścieniowej z repeaterem, która posiada wiele podstacji. Jeśli występuje retransmisja to trwa ona w kółko dzięki zamkniętemu obiegowi.

Projektowanie topologii sieci

Kolejnym problemem w komunikacji jest ograniczenie zasięgu sieci przez długość kabli lub zasięg transmisji. Potrzeba więc wykorzystać pośredników.

Przykładowe rozwiązania:

- Proste: połączenie każdego z każdym, ale nie jest to efektywna metoda.
- Lepsze: każdy z elementów posiada minimum 2 połączenia - w razie awarii jednej ze stacji ma alternatywę.

W projektowaniu topologii związkanych jest kilka istotnych zagadnień.

1 Uzgodnienie gotowości udziału w transmisji

1.1 Tryb połączeniowy

1.1.1 Idea rozwiązaania

Stacja A wysyła żądanie wyznaczenia trasy do stacji Z. Nazywamy to jako "pakiet żądania wykonania połączenia". Pakiet trafia do stacji B i jeśli ona może wysłać go dalej to odsyła potwierdzenie i szuka dalszej drogi. Fakt przesłania żądania zestawienia połączenia jest odnotowywany w każdym węźle przez który ten pakiet przechodzi (ID połączenia).

W końcu ten trafia do stacji Z i jeśli stacja Z się zgadza na połączenia to odsyła pakiet odpowiedzi pewną drogą (która wcale nie musi być zgodna z tą, którą przyszło żądanie) do stacji A. Dzięki temu, że w węzłach istnieją ID połączeń mamy wyznaczoną ścieżkę przez którą będą przebiegać komunikaty z A do Z, a nawet dwie.

1.1.2 Pakiet

ID połączenia	reszta
---------------	--------

1.1.3 Cechy trybu

W tym trybie nie ma problemu dublowania pakietów. Natomiast jeżeli jedno połączenie wysiądzie (np. na wskutek awarii stacji) to pojawia się poważny problem.

1.2 Tryb bezpołączeniowy

1.2.1 Idea rozwiązaania

Pakiet danych wysyłany wprost do sieci, niezależny przesył.

1.2.2 Pakiet

Adres odbiorcy	Adres nadawcy	reszta
----------------	---------------	--------

1.2.3 Cechy trybu

Tu mogą się zdarzyć duplikaty ramek itp. ale gdy któryś węzeł padnie to jest szansa, że wszystko dalej będzie działać.

W tym trybie działa większość sieci, w tym sam wielki Internet.

2 Tryb pracy sieci

2.1 Komutacja kanałów (*channel switching*)

Rezerwacja kanałów od Nadawcy do Odbiorcy. Na czas połączenia wszystkie połączenia od nadawcy do odbiorcy, włącznie z połączeniami pośrednimi, są zestawione i zablokowane. Jest związany z trybem połączeniowym.

2.2 Komutacja informacji (*information switching*)

Zamiast rezerwacji z góry wszystkich kanałów potrzebnych do komunikacji (komutacja kanałów) wyłącznie część między stacjami jest rezerwowana. Informacja jest przekazywana z węzła do węzła.

2.2.1 Przykład

Wysyłamy ze stacji A do C przez B. Ze stacji A przesyłamy wiadomość do B i tylko ten fragment kanału jest zajęty. Stacja B potwierdza odbiór i teraz ona zajmuje się znalezieniem C - połączenie pomiędzy stacjami A i B jest wolne. Tu komunikat musi być zapamiętany przez stację pośrednią. Jest to komunikacja typu **store & forward**

2.2.2 Problem

Wielkość informacji - każda stacja musi ją zapamiętywać.

Rozwiązania:

- **Komutacja wiadomości (information switching)**: zapis całości informacji w buforach w buforach itp. (chyba), limit 8kB, technologia ARPANET
- **Komutacja pakietów (packet switching)**: dzielenie informacji na pakiety o stałym rozmiarze.

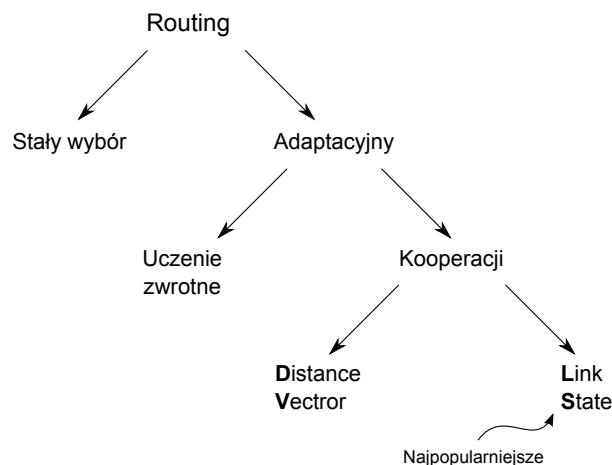
3 Adresacja

Wykorzystywana jest adresacja **IP**.

4 Wybór drogi (*routing*)

4.1 Diagram rozwiązań

Najpierw był routing źródłowy (*source routing*), czyli ręczne wyznaczanie drogi przez nadawcę. Następnie pojawiły się rozwiązania w postaci algorytmów.



4.2 Algorytmy stałego wyboru

Algorytm statyczny, drogę wyznacza administrator na sztywno.

4.3 Algorytmy adaptacyjne

Algorytm dynamiczny, każdy węzeł zbiera informacji potrzebne do wyznaczenia trasy. Każdy węzeł wyznacza z osobna gdzie iść dalej.

Z tym typem algorytmów związane są następujące zagadnienia:

4.3.1 Zdobywanie informacji

- **Uczenie zwrotne** - dodanie dodatkowego pola, gdzie przechowywane są informacje jak np. znaczek czasu wysłania pakietu. Dzięki temu węzeł wie jak długo szedł pakiet. następnie robi tabelkę. Np.

		Zi		
		B	C	K
Interfejs	et1	t_{Be1}		
	et2	t_{Be2}		
	et3	inf		

Na początku tablica jest pusta, co jest problemem. należy utworzyć początkową tablicę w jakiś sposób np. algorytmem stałego wyboru. Zamiast czasu, jako informację, lepiej przechowywać liczbę węzłów przez które się przehodzi, ponieważ aktualizacja czasu może powodować oznaczenie drogi jako niedostępna (?).

- **Algorytm kooperacji (samodzielnego badania (?)**) - każdy węzeł odpytuje otoczenie, jakie węzły sa obok niego, wysyła żądanie echa i otrzymuje odpowiedzi. Dzięki temu zna również czas. Echo musi czekać na obsługę jak każdy inny pakiet. Dzięki temu przesyłaniu (echo itd.) wychodzi metryka łącza.

Tutaj informacja ma swój czas życia ($30 - 60 \mu s$). Jeżeli żaden pakiet nie przehodzi ta drogą to znaczy, że coś jest z nią nie tak.

Rodzaje:

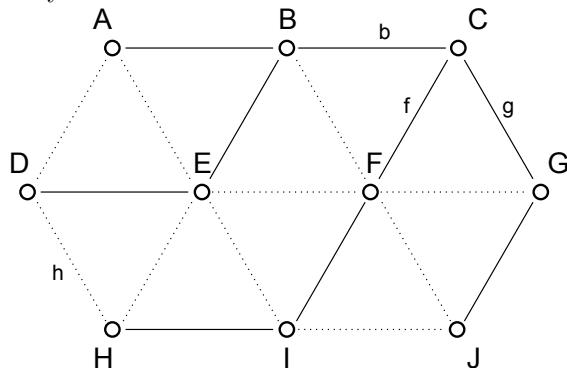
- *distance vector* - wysyła pełną tablice routingu do sąsiadów (drogi dostępne w otoczeniu i metryki)
- *link state* - przesyła wszędzie (?) informacje, że istnieje takie a takie łącze i metryka (koszt) tego łącza Np.

4.4 Inne algorytmy

- Zalewania (transmisji *broadcast*) - wysyłanie wszędzie w TTL
- Gorącego ziemniaka - wyślij do najbliższego / najlepszego z TTL

4.5 Wyznaczanie tablicy routingu

Przykładowa sieć.



Linie przerywane oznaczają brak drogi z jednej stacji do drugiej.

4.5.1 Graf drzewa spływu

Wyciąć te które mogą powodować zapętlenie i potem szuka się tych dróg o najmniejszym koszcie. Przykład z tablicą decyzyjną dla stacji C:

Routing table

Adres docelowy	Decyzja	Koszt
A	b	2
B	b	1
C	-	-
D	b	3
E	b	2
F	f	1
G	g	1
H	f	3
I	f	2
J	g	2

(*ce vector*).

Adres docelowy oraz decyzja tworzą wektor odległości (*distance vector*).

Kolumna decyzja to *routing*.

4.5.2 Algorytm stałego wyboru

Wykorzystuje tablicę stałego wyboru. Faworyzuje pewne drogi przed innymi.

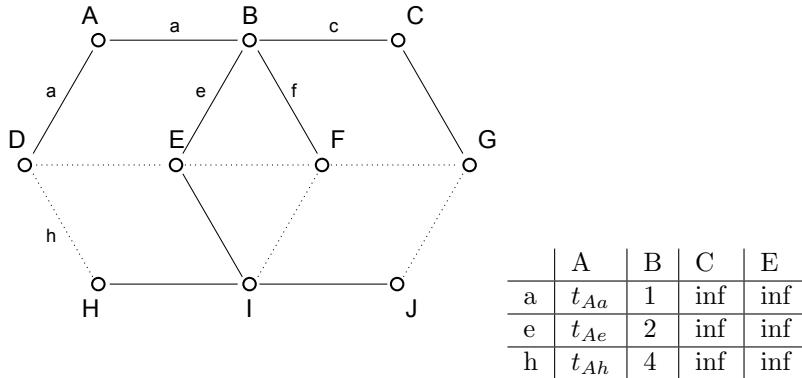
$r(x)$ - generator liczb losowych, który mówi, którą trasę wybrać. Jest to próba rozproszenia ruchu po sieci.

Routing table

	Decyzja 1	Decyzja 2	Decyzja 3
	$x \leq 0.7$	$0.7 < x \leq 0.9$	$x > 0.9$
A	b	f	g
B	b	f	g
C	-	-	-
D	a	e	c
E	a	e	c
F	f	g	b
G	f	g	b
H	f	b	g
I	f	b	g
J	f	b	g

Tabela posiada kolumnę z jedną główną trasą, która ma największe szanse się pojawić. Pozostałe dwie, mniej optymalne, są rezerwowe i mają mniejsze prawdopodobieństwo. Droga jest stała i ustalona zgórę (chyba).

4.5.3 Algorytm zwrotnego uczenia

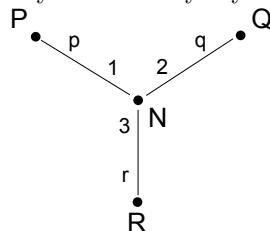


- t_{Aa} - przekazanie pakietu do D z A przez a.
- Algorytm nie potrafi wystartować od zera, dlatego na niemożliwe ścieżki wstawia się nieskończoność.
- Informacja jest stale aktualizowana. Mówi ona ile czasu potrzeba by wysłać pakiet różnymi ścieżkami.
- Przy przesyłce w drugą stronę wybiera się trasę najszysząszą. Nie musi być to pierwotna → jeśli istnieje inna szybsza, tedy idzie.
- Węzeł chce nauczyć się topologii sieci. Zapisuje z każdego przychodzącego pakietu informację o czasie przesyłu. Jeśli nic nie przychodzi to oznacza ścieżkę jako niedostępną.
- Węzeł nie posiada szczegółowych informacji o ścieżce, jedynie do którego sąsiada ona prowadzi.
- Udowodniono, że zestaw dróg lokalnie optymalnych powinien pokrywać się z trasą globalnie optymalną.

4.5.4 Algorytm kooperacyjny z *distance vector*

Znany jako algorytm Bellmana-Forda.

- Wymiana metryk tylko z najbliższymi sąsiadami.



Gdzie:

- Liczby to metryki
 - Duże litery to adresy
 - Małe litery to ścieżki

Można z tego ułożyć ładną początkową tablicę.

- Następnie węzeł N otrzymuje tablice od sąsiadów oraz dodaje sobie koszt dotarcia do sąsiadów.

P	
Adres	Metryka
A	3
C	5
D	7
R	4
S	7
Q	8

Q	
Adres	Metryka
B	3
D	5
R	3
S	5

+2

R	
Adres	Metryka
A	4
B	7
C	3
D	2
+3	

- W efekcie czego stacja N otrzymuje tablicę możliwości połączeń z innymi, dalszymi stacjami oraz wylicza sobie najkrótsze ścieżki (albo "interfeisy").

N		
Adres	Metryka	Ścieżka
A	4	p
B	5	q
C	6	p
D	5	r
P	1	p
Q	2	q
R	3	r
S	7	q

- Wymiana informacji w stałych odstępach czasu, 30 - 60 s, nie za szybko, nie za wolno.
 - Gdy koszt jest taki sam na dwóch i więcej trasach to teoretycznie można wybrać dowolną z nich.
 - Adres i metrykę nazywamy wektorem odległości, który jest wysyłany do sąsiadów.

5 Uszkodzenia w sieci

Wszystkie powyższe algorytmy działają w założeniu, że sieć jest sprawna oraz interesujące nas stacje działają i nie są uszkodzone. Jednak pojawia się kolejny problem: jeżeli któraś ze stacji zostanie uszkodzona i nie może przesyłać wiadomości, inne stacje muszą zostać o tym powiadomione. Analogicznie powinny być informowane o włączaniu stacji do sieci.

5.1 Rozchodzenie się informacji

5.1.1 Włączenie stacji

	A	B	C	D	E
0.	x	∞	∞	∞	∞
1.	0	1	∞	∞	∞
2.	0	1	2	∞	∞
3.	0	1	2	3	∞
4.	0	1	2	3	4

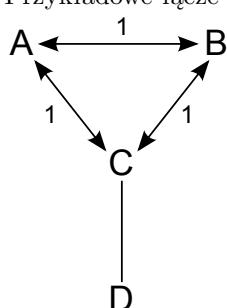
5.1.2 Odłączenie stacji

	A	B	C	D	E
0.	0	1	2	3	4
1.	x	3	2	3	4
2.	0	3	4	3	4
3.	0	5	4	5	4
4.	x	5	6	5	6
			...		
		∞	∞	∞	∞

Powoduje liczenie do nieskończoności.

5.2 Rozwiązywanie - Link State

- Zasada podziału horyzontów - określanie kierunków
- Nie są przesyłane niepotrzebne węzły (?)
- Przykładowe łącze



5.2.1 Stan łącza

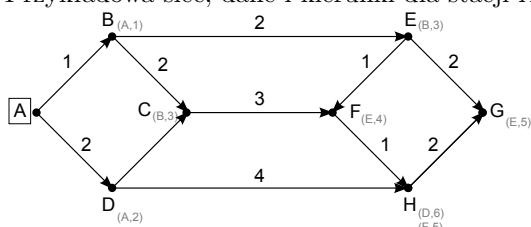
Nadawca	W
Nr sekwencyjny	
Wiek	
A	1
B	3

- Nr sekwencyjny - jeśli ktoś dostanie pakiet o większym numerze, to go przetwarza, a jeśli nie jest większy, to nie.
- Wiek - czas życia pakietu (w cyklu). Jeśli czas życia upływa to węzeł usuwa węzeł W ze swojej tablicy routingu.

5.2.2 Etapy Link State

1. "Zapoznaj się z sąsiadami" - wysłanie pakietu HELLO, który zawiera identyfikację nadawcy, oraz czekanie na takie same.
2. "Wyznacz metrykę trasy" - Echo Request & Echo Response
3. Algorytm dystrybucji pakietów LS

- Tworzenie tablicy decyzyjnej routingu (algorytm Dijkstry)
Przykładowa sieć, dane i kierunki dla stacji A.



Jeżeli mamy 2 różne wartości, np. $A \rightarrow D - 3$, $A \leftarrow D - 2$, to przypisujemy średnią.

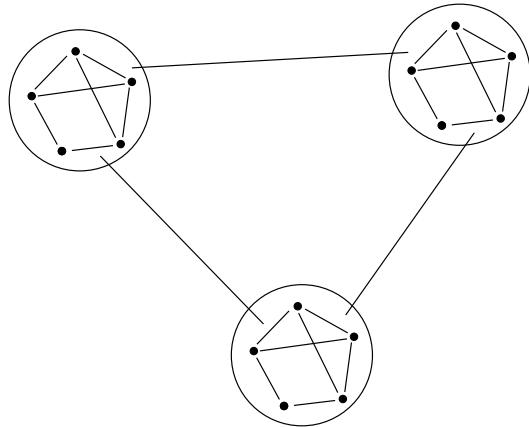
- Tablica routingu

Adres	Droga
B	b
C	b
D	d
E	b
F	b
G	b

- Zmiany w topologii są rozsyłane natychmiast - nie ma liczenia do nieskończoności.

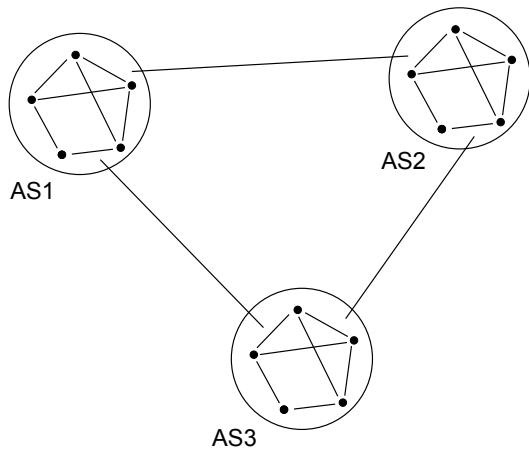
6 Problemy z liczbą węzłów

6.1 Metoda hierarchii



Jest tylko 1 wpis na całą sieć w tabeli routingu.
Zawiera wpisy na połączenia lokalne.

6.2 Inny sposób



System autonomiczny. AS1 - AS3 (mogą to być firmy, instytucje itp.) traktujemy jako osobne węzły, więc rysunek można uprościć do postaci zwykłego trójkąta.

6.3 Routery

Dzielimy je na:

- Zewnętrzne - EGP
- Wewnętrzne - IGP

Protokoły

0.4 IPv4

- A = 10.000 / 8

- 16-B 172.16 - 31.255.255 / 12
- B 192.168.0.0 / 16

0.5 Transport

IP	Port (16)
----	-----------

APIPA *Automatic Private IP Assigned* - algorytm uruchamiany, kiedy system nie może pobrać adresu z sieci, a ma pobrać adres z sieci. 169.254.0.0 - generowany jest losowy adres, który przepisywany jest systemowi. Komputery, które są odłączone od sieci automatycznie konfiguruują się tym algorytmem. Kiedy komputery nie mogły się uruchamiać bez adresu.

Broadcast, przykład:

27	5
----	---

- 00000 - Network
- 11111 - Broadcast

0.6 IPv6

Rozmiar 128 bitów, znacznie więcej niż w IPv4.

ver	PRID	Etykieta przepływową
Długość danych	Next Header	HopLimit
SourceAddress (128 bit)		
Destination Address		

1: 0, 2, 24, 31

2: 8, 8, 8

Next headers:

- Hop by hop options
- Routing
- Fragmentation
- Authentication
- Destination options
- Encryption security payload
- Jumbo header - nagłówek sygnalizujący specjalny rozmiar pakietu, 1 MB.

Routing

Next Header określa rodzaj.

Jumbo

Dla środowiska superkomputerów RFC 1883 - 1887.

- Unicast address - do jednego
- Multicast address - do wielu
- Anycast address - do jednego z wielu (w obrębie grupy)

2001:cdba:0000:0000:0000:3257:9652 - pełny 128-bitowy adres IP.

CDBA - heksadecymalnie, określają 4 grupy.

W zapisie grupa 4 zer może być zapisana jednym zerem.

2001:cdba:0:0:0:3257:9652

2001:cdba::3257:9652

Next header	Pole związane z nagłówkiem	Kod opcji (ilość adresów + network adres)
x	BitMap	
1 do 24 Adres IPv6		

Next header	Pole związane z nagłówkiem	194	
	Jumbo payload size		

0.6.1 Specjalne adresy IPv6

- **::/96** - adres zgodny z zapisem IPv4
- **::/128** - odpowiednik adresu IP składającego się z samych zer i oznacza sieć nieokreślona (*Unspecified Address*).
- **::1/128** - adresem local loopback jest adres składający się z samych zer i jednej jedynki.
- **2001:db8::/32** - *documentation prefix*
- **fec0::/10** - *site local prefix* - dla pierwszych 16 bitów, jeżeli są one w tym, to definiuje adresy używane wewnątrz danej lokalizacji.
- **fc00::/7** *unicast local address*
- **ff00::/8**
- **ff80::/10** - adres przydzielony w łączu (*link local prefix*)

0.7 Protokoły

0.7.1 TCP

Source IP				
Destination IP				
Source port		Destination port		
Nr sekwencyjny pakietu				
nr potwierdzenia pakietu				
TCP header length	x	Flagi	Window Size	
Suma kontrolna			Urgend pointer	
OPCJA				
SAC				

0.7.2 UDP

Flagi i komendy UDP:

- URG
- ACK
- PSH
- RST
- SYN
- FIN

Algorytmy, które wymuszają pewne mechanizmy zapobiegające przeciążeniu sieci. odpowiadają za zabezpieczenie transportu z warstwy trzeciej do warstwy czwartej - gubieniu pakietów.

Source port	Destination port
Packet length	Suma kontrolna

- slow start
- congestion avoidance
- multiplicative decrease

Obrazek: slow start Uzgodniony rozmiar okna (z odbiorca), którego nie możemy przekroczyć.