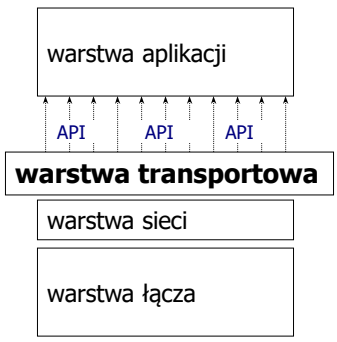
Warstwa transportu

Warstwa transportowa

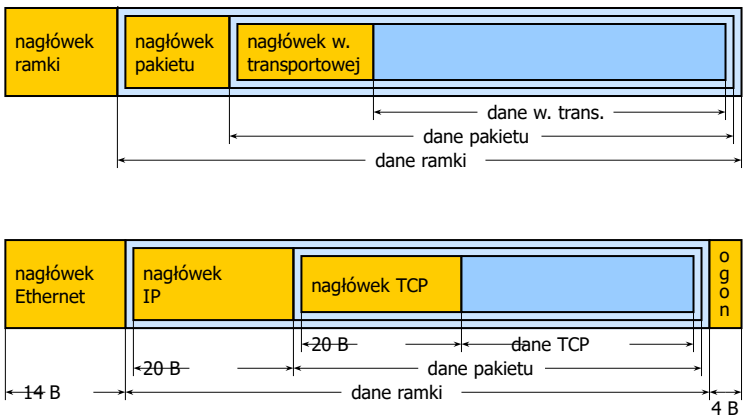
- Warstwa 4 modelu OSI
- Warstwa 3 modelu TCP/IP



Warstwa transportowa

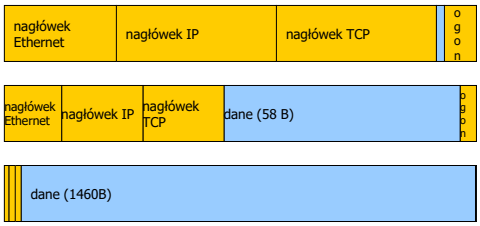
- Zapewnienie niezawodnego przesyłania danych /wg ISO/
- Transmisja połączeniowa
  - nawiązywanie połączenia
  - uzgadnianie parametrów połączenia
  - wysyłanie danych wysokopriorytetowych
- Transmisja bezpołączeniowa

Enkapsulacja



Enkapsulacja

- Narzut enkapsulacji może mieć duży wpływ na wielkość generowanego strumienia danych
- Przykładowo dla enkapsulacji Ethernet/IP/TCP oraz strumienia danych **64 kb/s**:



	Rozmiar segmentu TCP	Narzut nagłówków	Wymagane pasmo dla sieci
	1 B	5800%	~ 3,8 Mb/s
	58 B	100%	128 kb/s
	1460 B	~ 4%	~ 66 kb/s

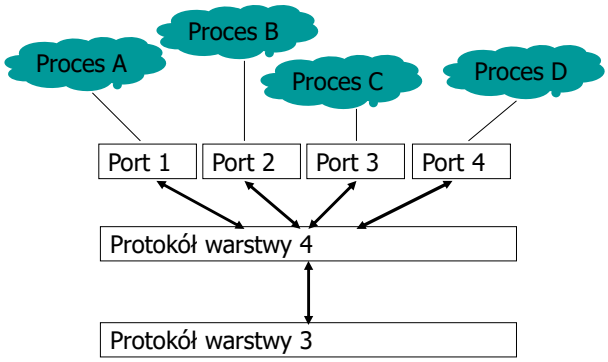
Porty

- Właściwie każdy współczesny komputer jest (może być) wyposażony w system operacyjny pozwalający na uruchamianie wielu zadań jednocześnie
- Identyfikacja procesów: numery
  - problemy:
    - numery procesów zmieniają się np. przy restarcie,
    - zmiana numerów punktów docelowych musiałaby być sygnalizowana innym hostom,
    - ...
  - rozwiązanie: schemat adresacji procesów wewnątrz każdego hosta

Port: abstrakcyjny punkt docelowy identyfikowany za pomocą dodatniej liczby całkowitej

Porty

- Multipleksacja / demultipleksacja segmentów



Porty

- Dostęp do portów może być:
  - synchroniczny: przetwarzanie jest wstrzymane na czas komunikacji,
  - asynchroniczny: bez przerywania przetwarzania
- Zazwyczaj dane przychodzące na dany port są buforowane. Dzięki temu:
  - dane przychodzące na port zanim proces obsługujący może je przetworzyć nie są tracone,
  - czas oczekiwania procesu na dane może zostać skrócony.
- UWAGA! Bufory mają ograniczoną długość!

Numery portów dla różnych protokołów warstwy 4 mogą się powtarzać.

Porty - programowanie

- Asocjacja jest to piątka:
  (protokół, adres lokalny, proces lokalny, adres obcy, proces obcy)
- Półasocjacja inaczej **gniazdo** (socket)
  to trójka:
  (protokół, adres lokalny, proces lokalny)
  lub
  (protokół, adres obcy, proces obcy)
- W przypadku protokołów TCP i UDP procesy identyfikowane są przez numery portów określone w nagłówku

Porty

- Zestaw **dobrze znanych** portów określa porty, z którymi mogą łączyć się klienci (RFC 1700) np.:
  - 21 FTP
  - 23 Telnet
  - 25 Mail (SMTP)
  - 80 HTTP
- Porty **efemeryczne** — krótkotrwałe, określane przez moduł warstwy transportowej klienta na czas połączenia

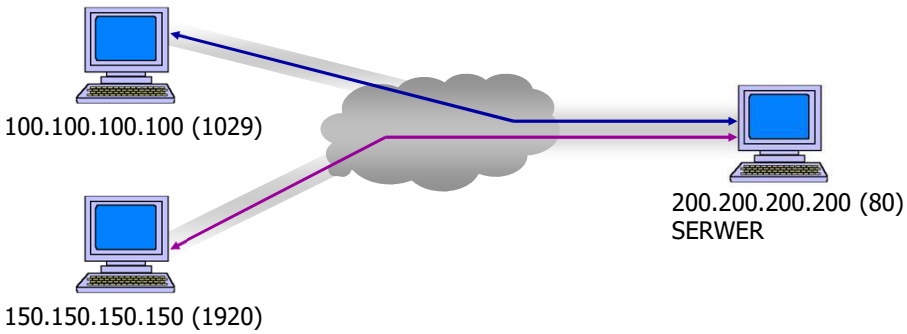
Porty

- Możliwe jest więcej, niż jedno połączenie na port. Trudno sobie wyobrazić, by serwer SMTP zmieniał port dla każdego połączenia

Usługa docelowa może obsługiwać jednocześnie wielu klientów, gdyż dialog między nimi (połączenie) jest identyfikowany przez końce połączeń (ang. endpoints).

Pozwala to różnym klientom na jednoczesne łączenie się na ten sam port serwera realizującego daną usługę.

Porty a połączenia



	Protokół	Adr. lok.	Proc. lok.	Adr. zd.	Proc. zd.
Połączenie 1:	(TCP,	200.200.200.200,	80,	100.100.100.100,	1029)
Połączenie 2:	(TCP,	200.200.200.200,	80,	150.150.150.150,	1920)

Interfejs programisty

- Interfejsy niskiego poziomu
  - gniazda BSD (UNIX, Linux)
  - Transport Layer Interface – TLI (UNIX, Linux)
  - winsock (Microsoft)
- Interfejsy wysokiego poziomu
  - java.net (Sun)
  - biblioteka MFC (Microsoft)

Rodzaje protokołów

- Połączeniowy
  - nawiązywanie połączenia
  - transmisja danych
  - zamykanie połączenia

Większe możliwości (zapewnienie, że dane dotrą do celu itp.)

- Bezpołączeniowy
  - tylko transmisja...

Mniejsze narzuty na informacje kontrolne

Transmission Control Protocol

- Oczekiwanie warstw wyższych (i programistów :-)):
  - możliwość przesyłania strumieni danych bez konieczności
    wpisywania w każdy program obsługi błędów
    - na maszynie odbiorcy ma się pojawić dokładnie taki sam ciąg oktetów, jak wysłany przez nadawcę

Transmission Control Protocol

- Połączeniowy
  - nawiązywanie połączenia
    (zestawienie obwodu wirtualnego)
    - złudzenie istnienia obwodu jest realizowane za pomocą mechanizmów kontrolnych TCP,
    - procedura *three-way handshake.*
  - zamykanie połączenia
  - każde połączenie ma dokładnie dwa końce
  - za pomocą połączenia możliwa jest transmisja w obie strony (full-duplex)
    - przy okazji do danych da się „dokleić" informacje kontrolne
  - połączenia są buforowane w sposób niewidoczny dla aplikacji (powód: narzuty enkapsulacji)

Transmission Control Protocol

- Niezawodny
  - potwierdzenia odbioru (ang. positive acknowledgement)
  - retransmisje
  - kontrola przepływu danych
  - suma kontrolna dla nagłówka i danych
  - porządkowanie kolejności segmentów
  - usuwanie segmentów zdublowanych

Transmission Control Protocol

- Co specyfikuje protokół TCP?
  - format danych,
  - procedury inicjalizacji i zamknięcia połączeń,
  - procedury upewnienia się, że dane dotarły,
  - sposób rozróżniania odbiorców,
  - sposoby reagowania na błędy
- Czego protokół TCP nie specyfikuje?
  - interfejsu programisty (API)

Nagłówek TCP

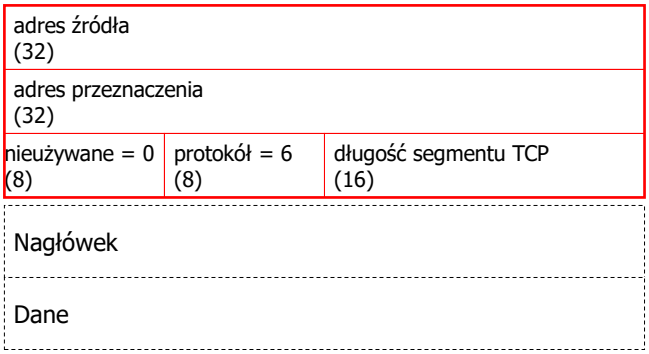
- Nagłówek ma rozmiar 20 bajtów + opcje

Flagi TCP

- SYN synchronizacja numerów sekwencyjnych (ang. Initial Sequence Number — ISN)
- ACK pole potwierdzenia (ang. *acknowledgment)* zawiera aktualny numer potwierdzenia
- FIN zakończenie (ang. *finish)* przesyłania danych
- RST zresetowanie (ang. *reset*) połączenia
- URG dane pilne (ang. *urgent)* począwszy od wskaźnika ważności
- PSH przekazanie danych do aplikacji najszybciej jak to możliwe (ang. *push*)

Suma kontrolna

- Obliczanie sumy kontrolnej dla nagłówka, danych oraz pseudonagłówka



Suma kontrolna

- Do czego służy jeszcze jedna suma kontrolna?
  - warstwa 2: poprawność ramki, ew. korekta,
  - warstwa 3: poprawność nagłówka pakietu,
  - warstwa 4: poprawność pseudonagłówka, nagłówka segmentu oraz przesyłanych danych
- Pseudonagłówek pozwala na zweryfikowanie, czy ramka dostarczona przez warstwę 3 rzeczywiście jest adresowany do danego odbiorcy
- Uwaga: ten sposób obliczania sumy kontrolnej oznacza, że protokół warstwy 4 korzysta z danych warstwy 3, co nie do końca jest zgodne z modelem OSI

Opcje

- Maksymalny rozmiar segmentu (ang. Maximum Segment Size MSS)
  - im większe segmenty tym wydajniejsze przesyłanie danych
  - należy unikać fragmentacji pakietów IP
  - duże segmenty mają niekorzystny wpływ np. na transmisje danych w czasie rzeczywistym
- Skalowanie okna
- Znacznik czasowy — określanie RTT

Przykład połączenia

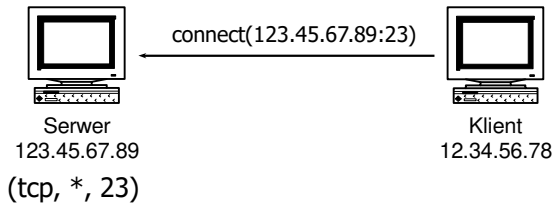
- Serwer tworzy gniazdo np. (tcp, *, 23) i oczekuje na połączenie od klienta (pasywne otwarcie)

Serwer
123.45.67.89
(tcp, *, 23)

Klient
12.34.56.78

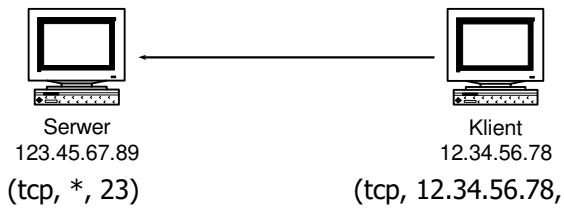
Przykład połączenia

- Klient łączy się z serwerem, otwierając połączenie poleceniem **connect** (aktywne otwarcie)



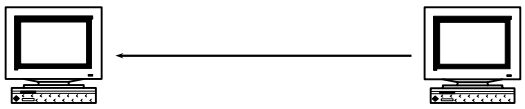
Przykład połączenia

- Moduł TCP tworzy gniazdo klienta z efemerycznym numerem portu np. 1234



Przykład połączenia

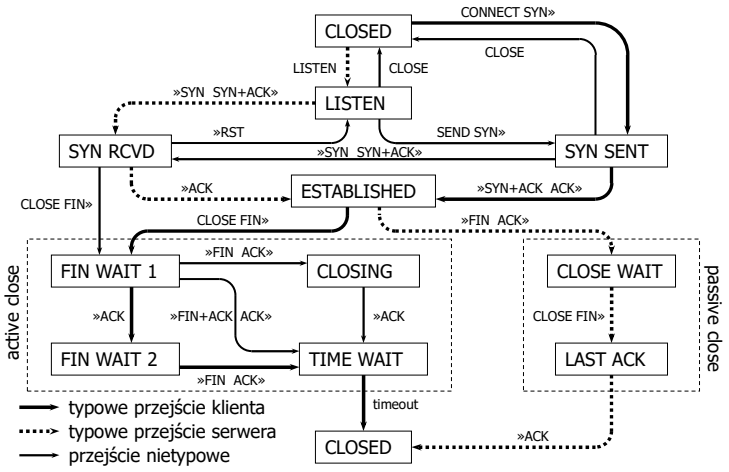
- Serwer otrzymuje zgłoszenie od klienta
- Moduł TCP tworzy unikalną asocjację:
  (tcp, 123.45.67.89, 23, 12.34.56.78, 1234)



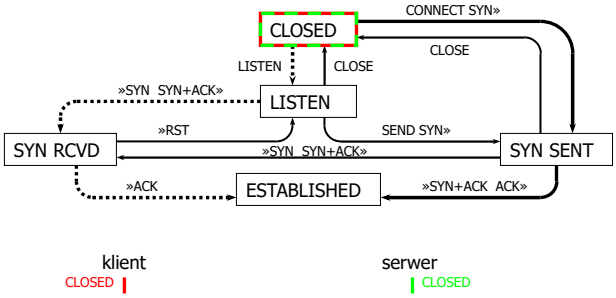
Serwer
123.45.67.89
(tcp, 123.45.67.89, 23)

Klient
12.34.56.78
(tcp, 12.34.56.78, 1234)

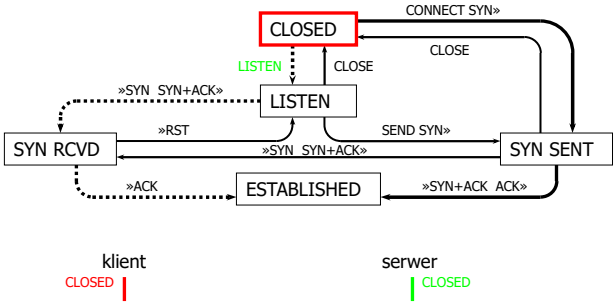
Schemat stanów TCP



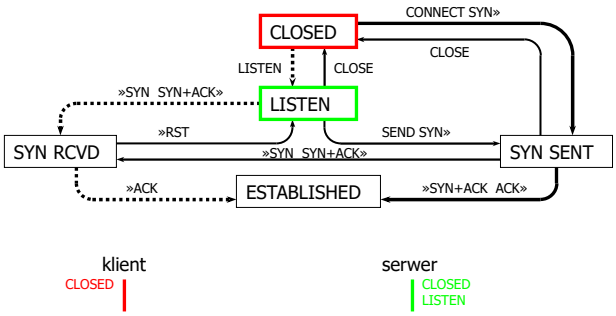
Nawiązanie połączenia



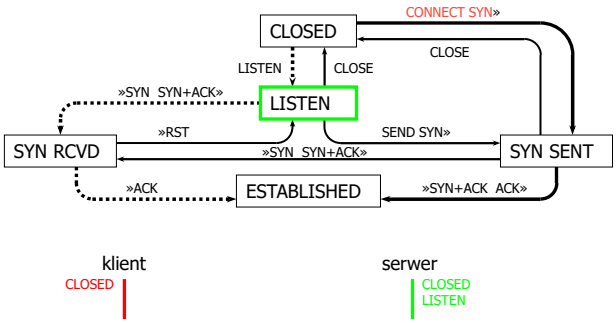
Nawiązanie połączenia



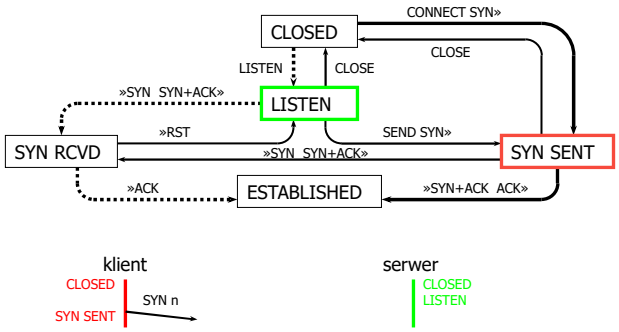
Nawiązanie połączenia



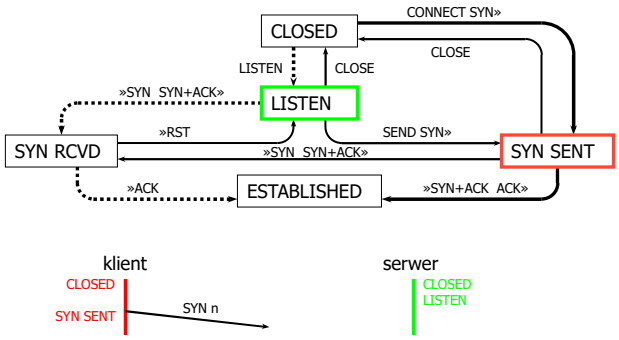
Nawiązanie połączenia



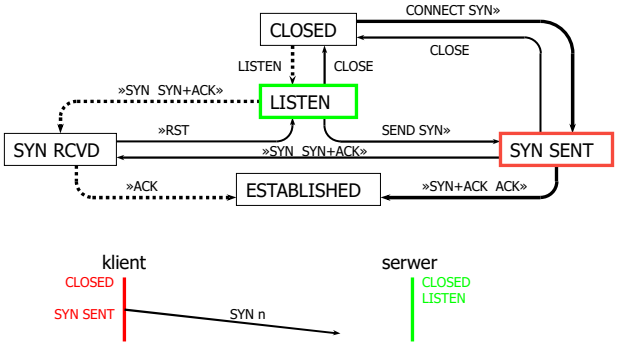
Nawiązanie połączenia



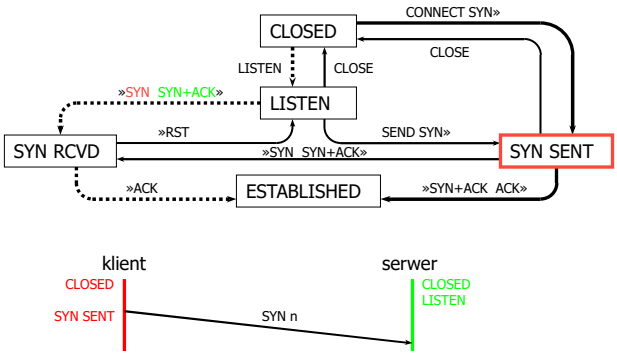
Nawiązanie połączenia



Nawiązanie połączenia



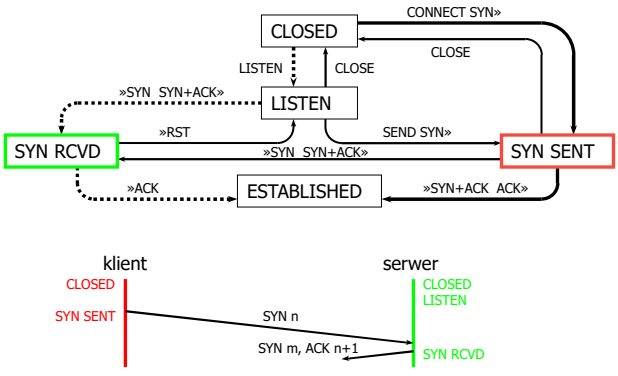
Nawiązanie połączenia



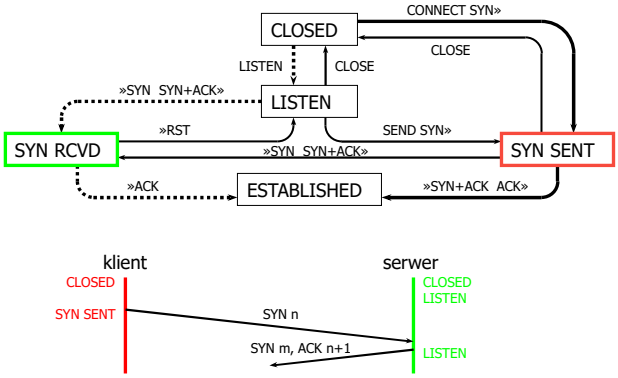
Nawiązanie połączenia

- Klient wysyła do serwera segment SYN wraz z informacją o dolnej wartości numerów sekwencyjnych (n) używanych do numerowania segmentów wysyłanych przez klienta (np. 100) a następnie przechodzi w stan SYN SENT,

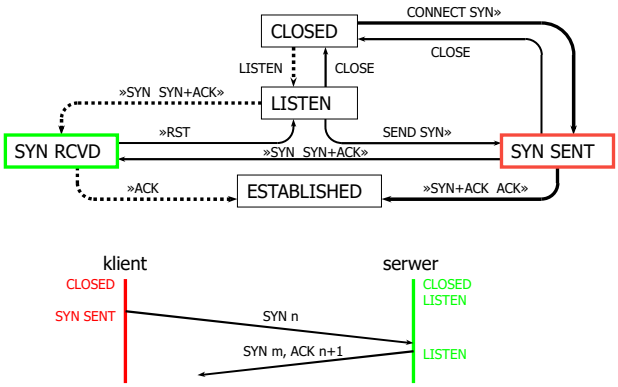
Nawiązanie połączenia



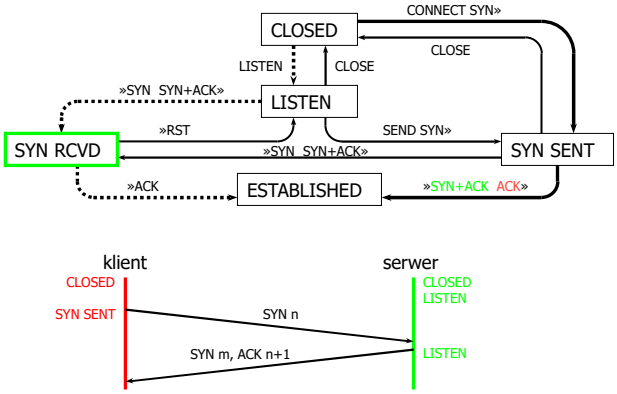
Nawiązanie połączenia



Nawiązanie połączenia



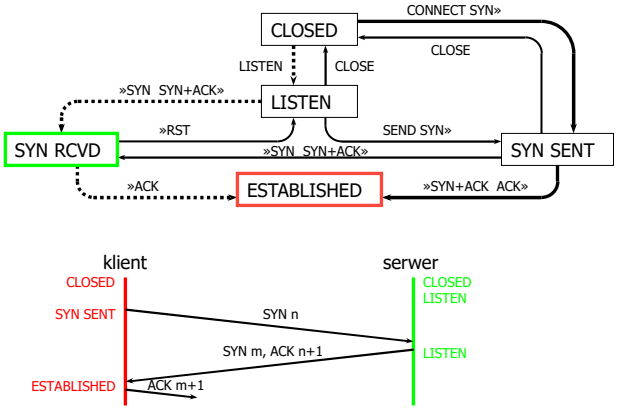
Nawiązanie połączenia



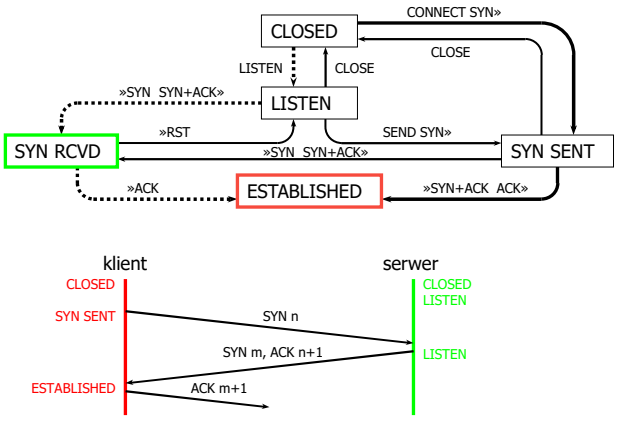
Nawiązanie połączenia

- Serwer, po otrzymaniu segmentu SYN, przechodzi w stan SYN RCVD i jeśli chce również nawiązać połączenie to wysyła klientowi segment SYN z informacją o dolnej wartości numerów sekwencyjnych (m) używanych do numerowania segmentów wysyłanych przez serwer (np. 300) oraz segment ACK z polem numeru sekwencji ustawionym na wartość o jeden większą niż wartość pola sekwencji pierwszego segmentu SYN klienta (n + 1, np. 101).

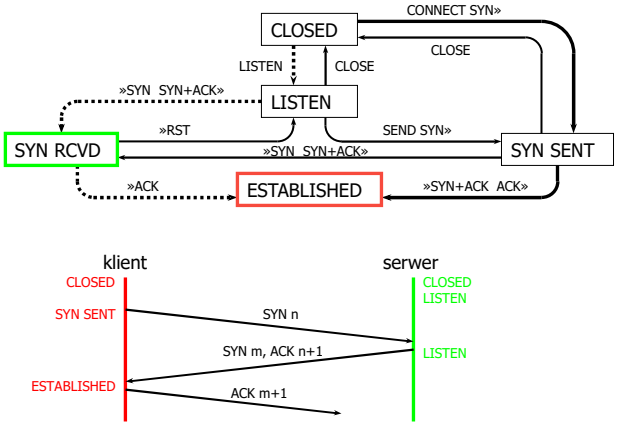
Nawiązanie połączenia



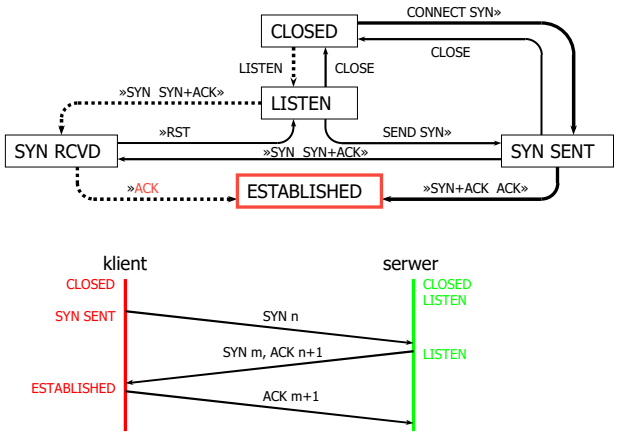
Nawiązanie połączenia



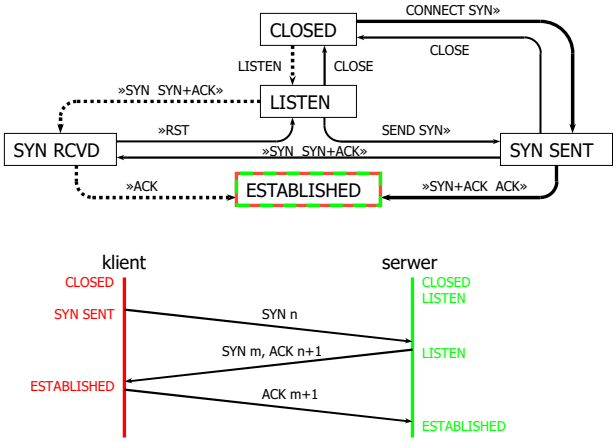
Nawiązanie połączenia



Nawiązanie połączenia



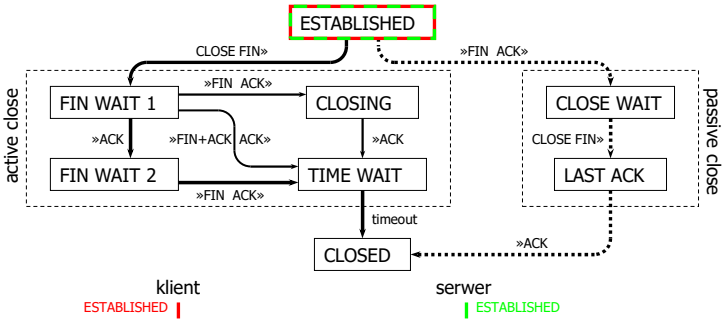
Nawiązanie połączenia



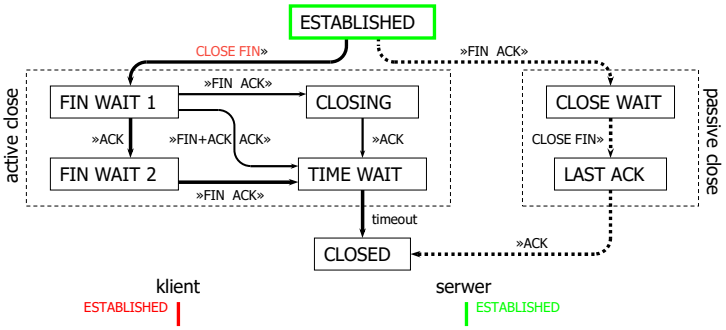
Nawiązanie połączenia

- Klient, po odebraniu segmentów SYN i ACK od serwera przechodzi w stan ESTABLISHED i wysyła do niego segment ACK potwierdzający odebranie segmentu SYN (numer sekwencji ustawiony na m+1, np. 301)
- Serwer odbiera segment ACK i przechodzi w stan ESTABLISHED
- Klient może teraz rozpocząć przesyłanie danych

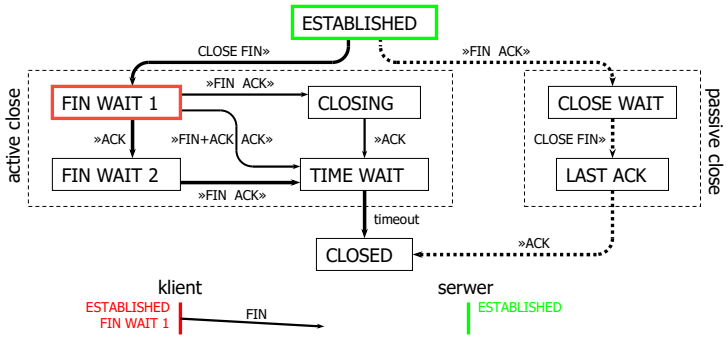
Zamknięcie połączenia



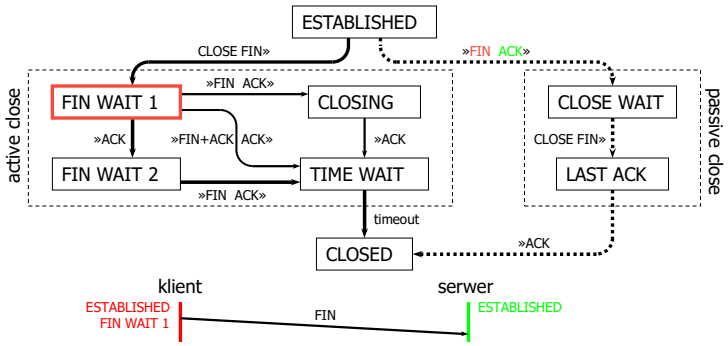
Zamknięcie połączenia



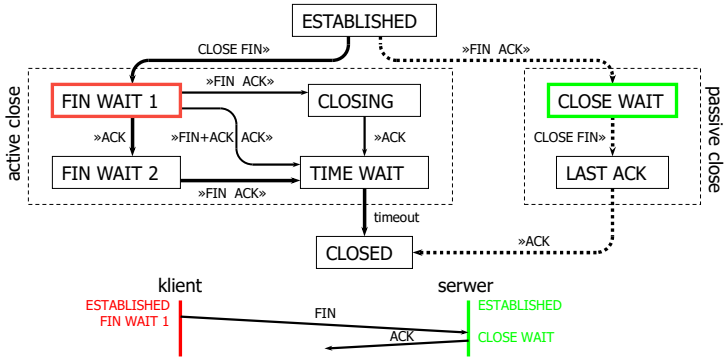
Zamknięcie połączenia



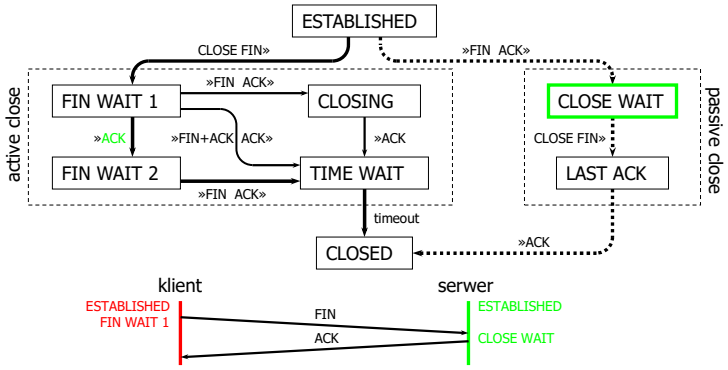
Zamknięcie połączenia



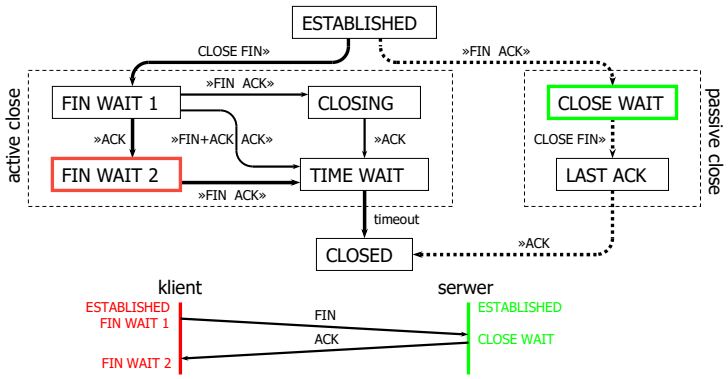
Zamknięcie połączenia



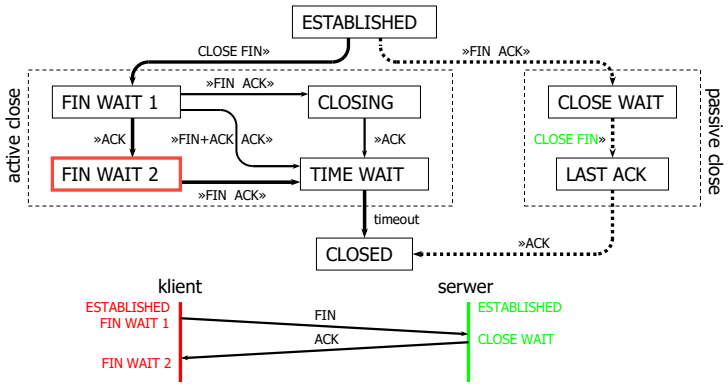
Zamknięcie połączenia



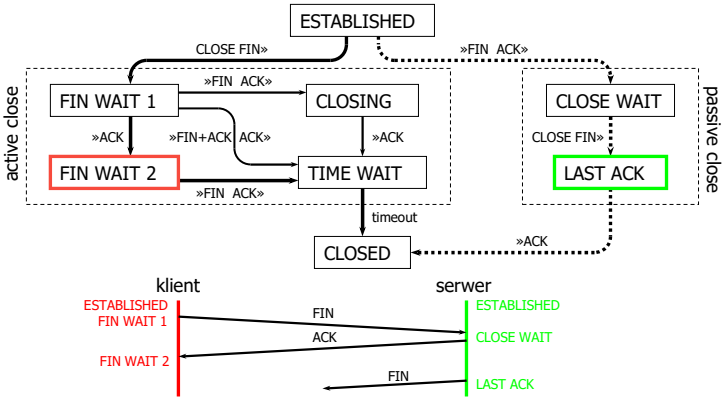
Zamknięcie połączenia



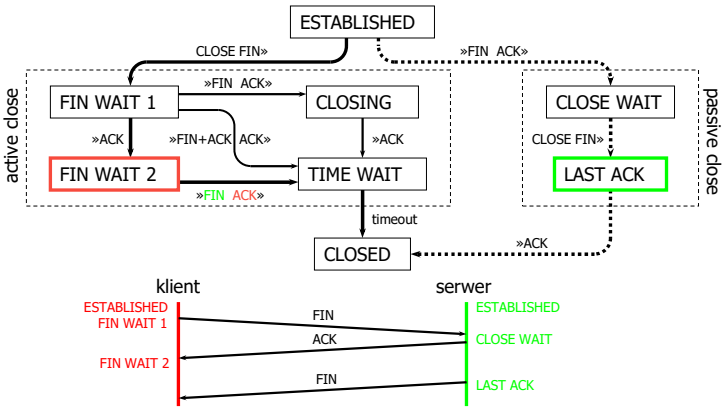
Zamknięcie połączenia



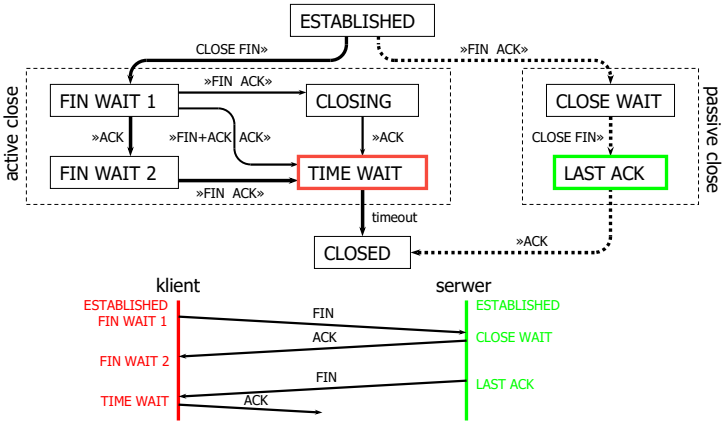
Zamknięcie połączenia



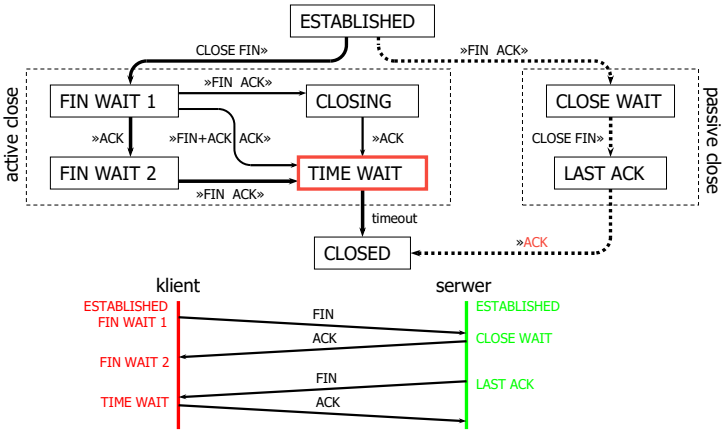
Zamknięcie połączenia



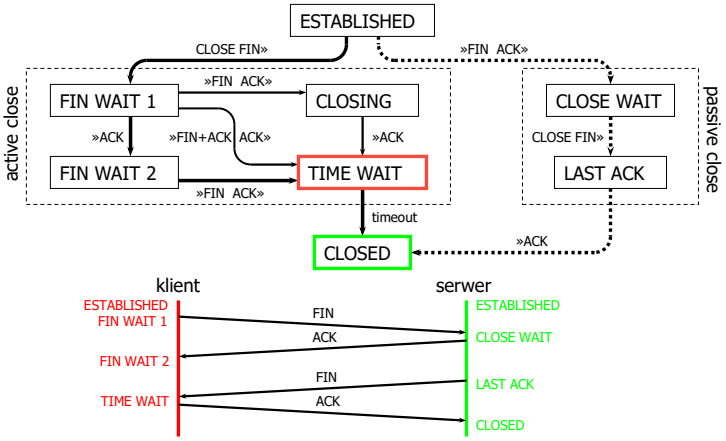
Zamknięcie połączenia



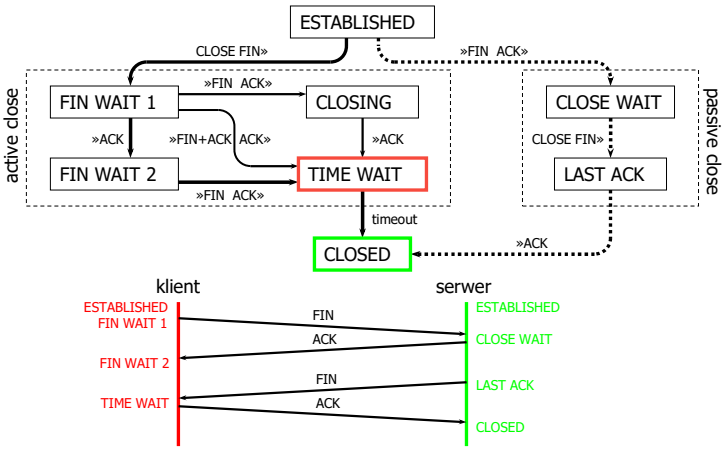
Zamknięcie połączenia



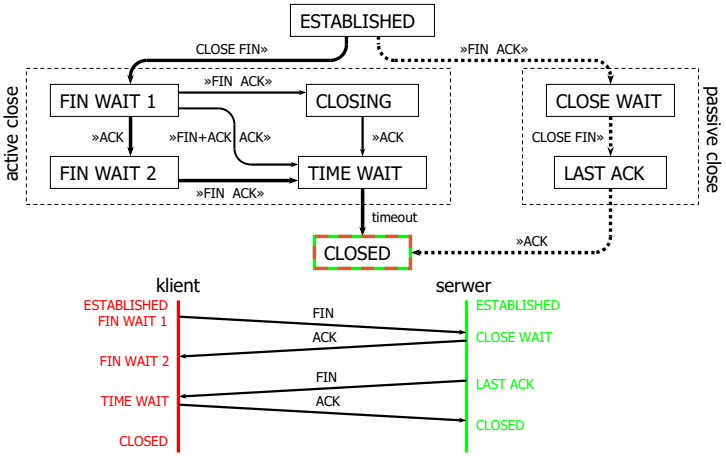
Zamknięcie połączenia



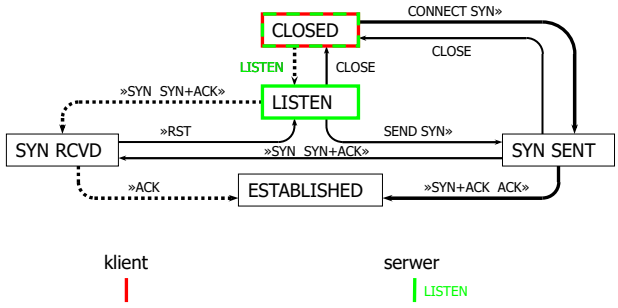
Zamknięcie połączenia



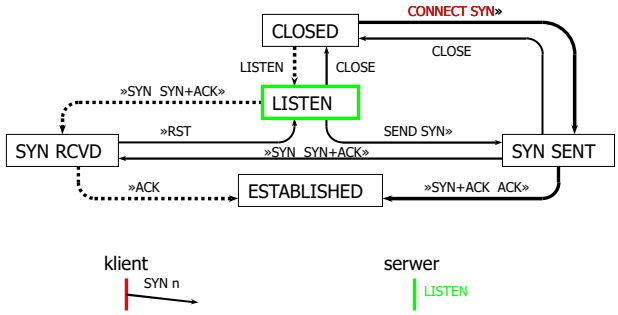
Zamknięcie połączenia



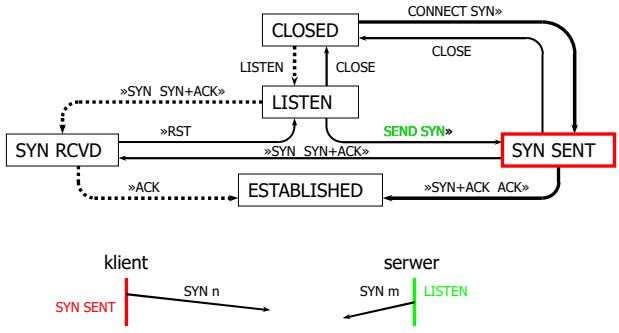
Jednoczesne otwarcie



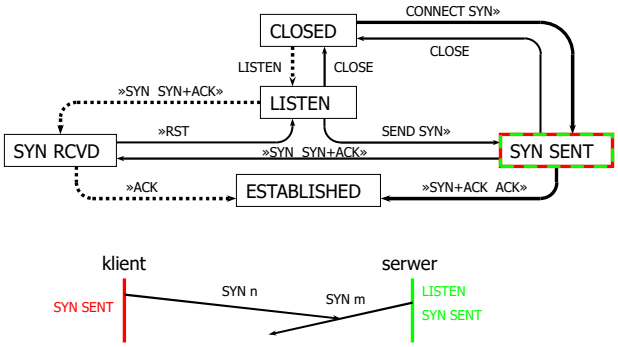
Jednoczesne otwarcie



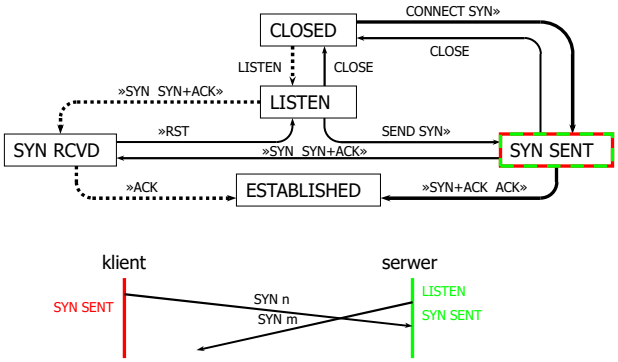
Jednoczesne otwarcie



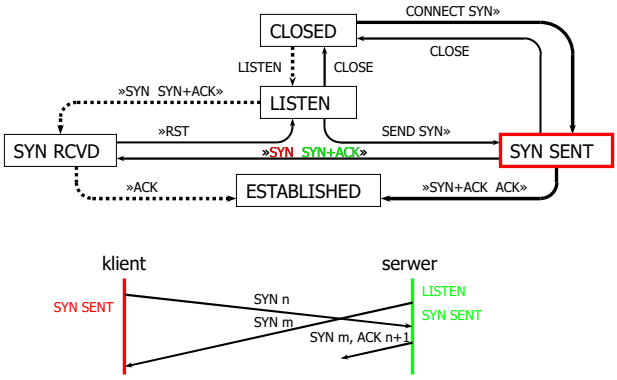
Jednoczesne otwarcie



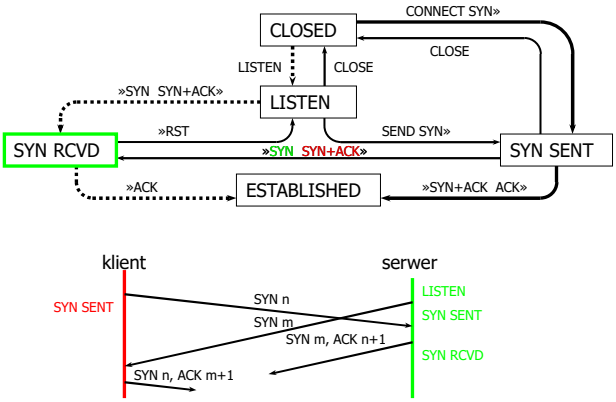
Jednoczesne otwarcie



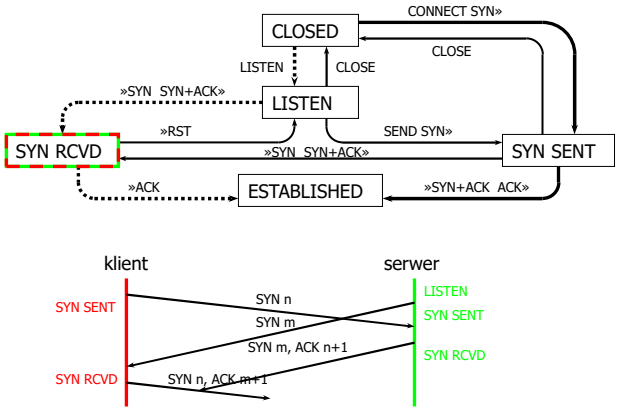
Jednoczesne otwarcie



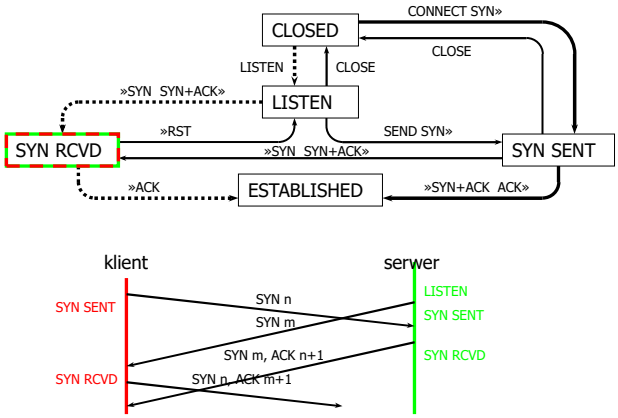
Jednoczesne otwarcie



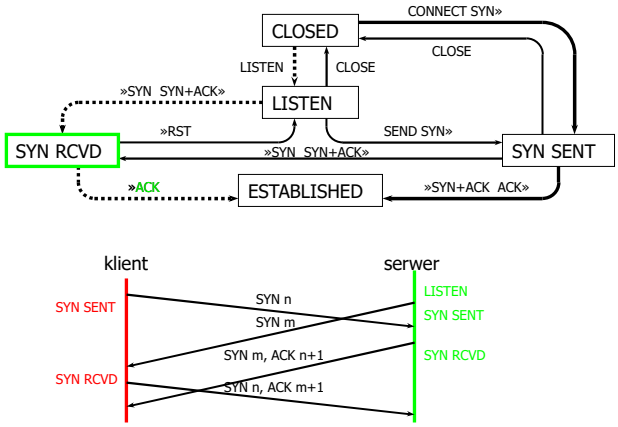
Jednoczesne otwarcie



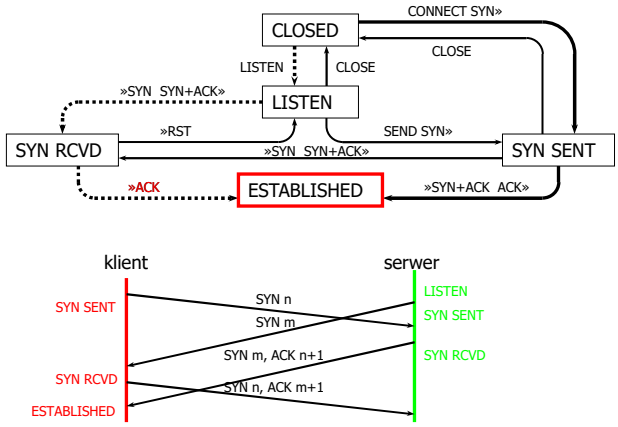
Jednoczesne otwarcie



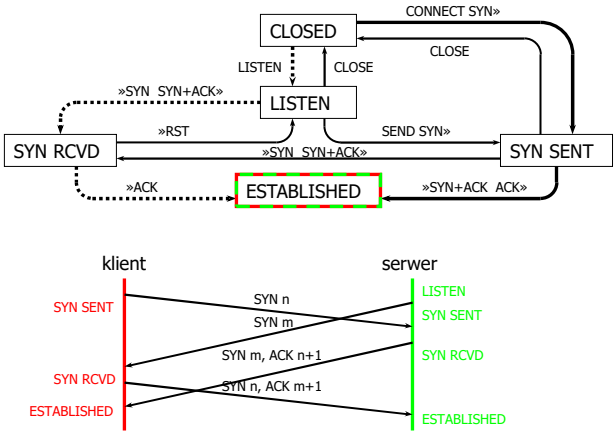
Jednoczesne otwarcie



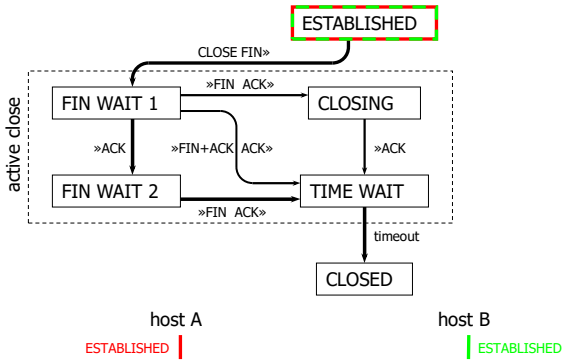
Jednoczesne otwarcie



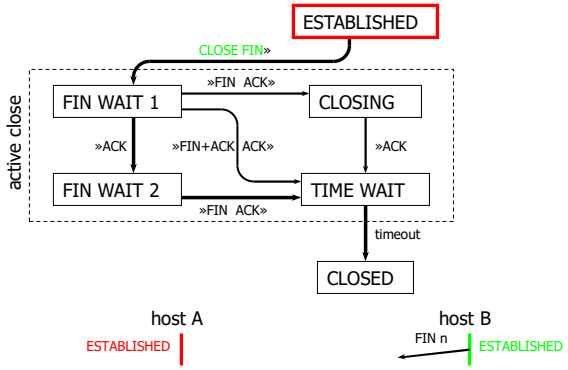
Jednoczesne otwarcie



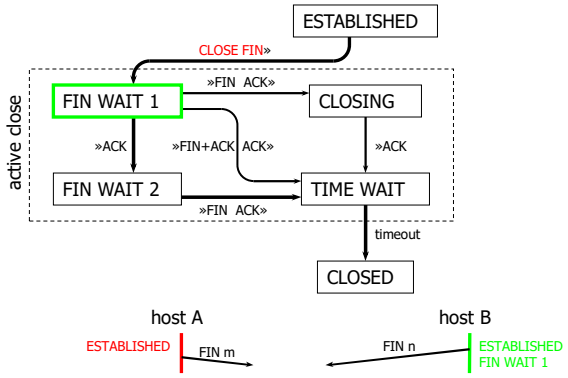
Jednoczesne zamknięcie



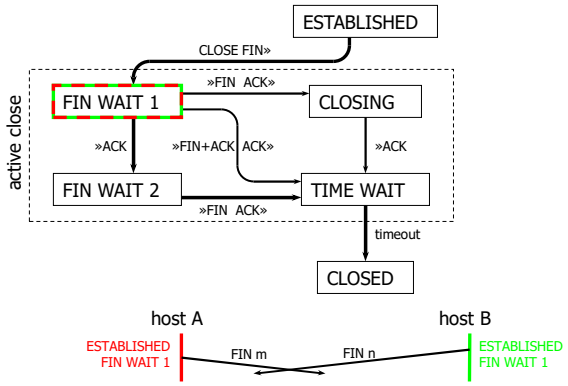
Jednoczesne zamknięcie



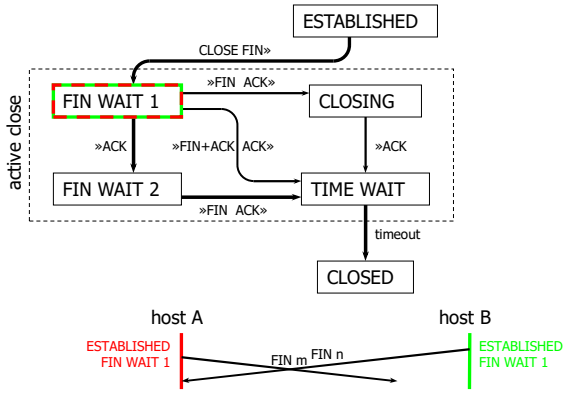
Jednoczesne zamknięcie



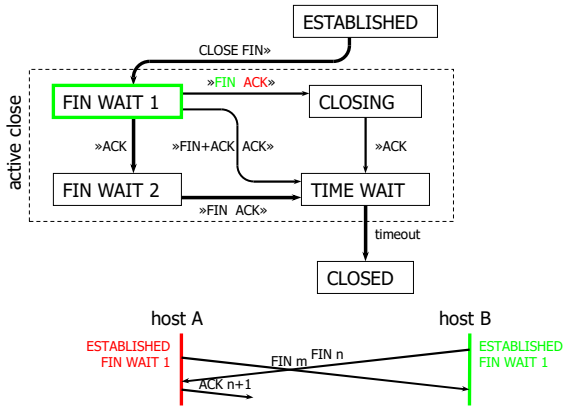
Jednoczesne zamknięcie



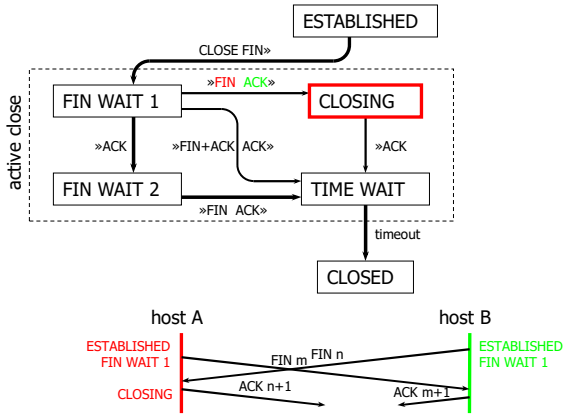
Jednoczesne zamknięcie



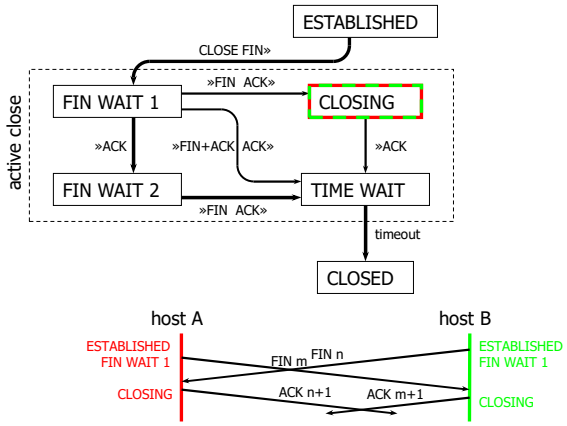
Jednoczesne zamknięcie



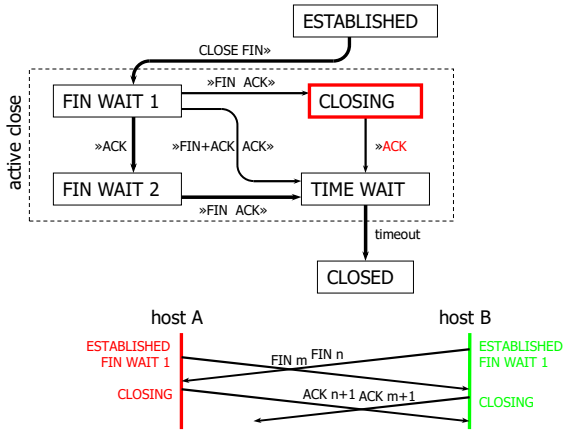
Jednoczesne zamknięcie



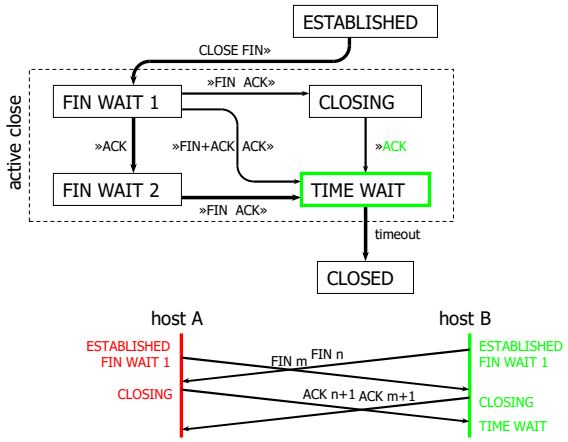
Jednoczesne zamknięcie



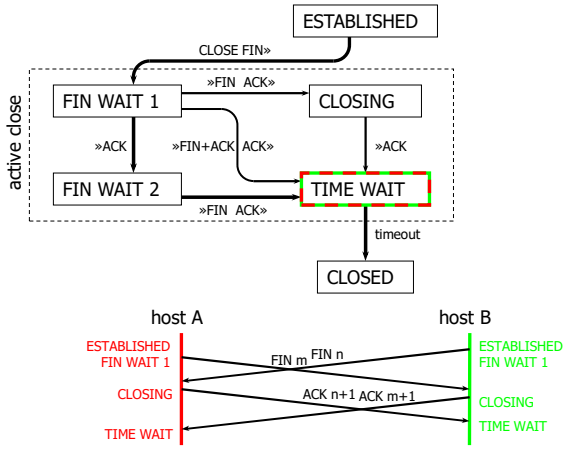
Jednoczesne zamknięcie



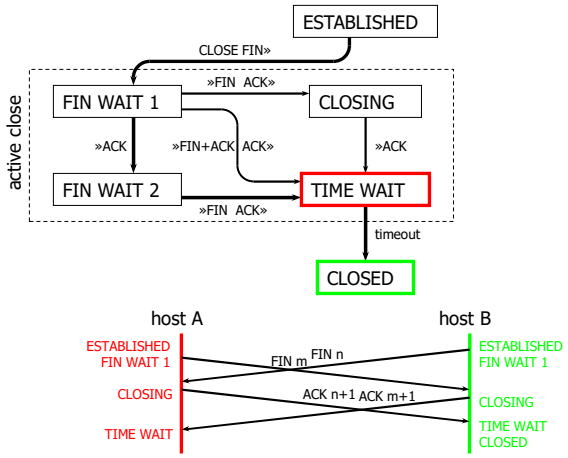
Jednoczesne zamknięcie



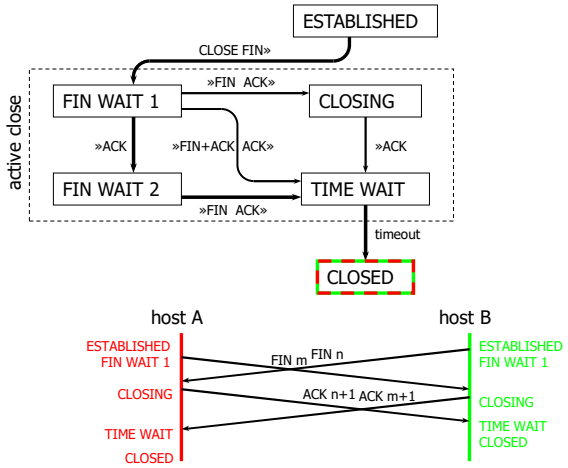
Jednoczesne zamknięcie



Jednoczesne zamknięcie



Jednoczesne zamknięcie



Przesyłanie danych

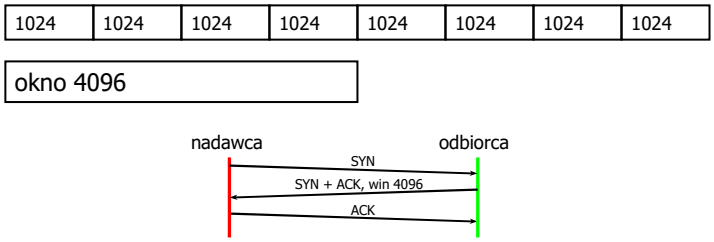
- Przesyłanie danych masowych
  - przesuwane okna
  - powolny start
- Przesyłanie danych interaktywnych
  - algorytm Nagle'a
  - opóźnione potwierdzenie
- Eliminacja syndromu głupiego okna

Przesuwane okno

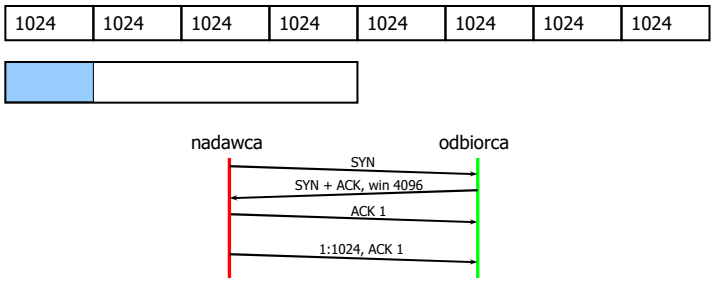
- Zapobiega przeciążeniu odbiorcy
  - odbiorca określa ile danych jest w stanie przyjąć
  - szybki nadawca musi poczekać, aż odbiorca otworzy dla niego okno
- Zasada działania
  - ogłoszenie okna
  - zamykanie (wysyłanie danych)
  - otwieranie (odbieranie danych)
  - kurczenie się (zabronione)

Rozmiar okna przesyłany jest w każdym potwierdzeniu i określa stan zapełnienia bufora odbiorcy

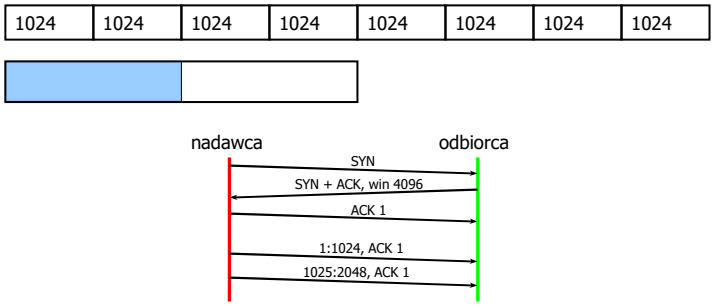
Przesuwane okno



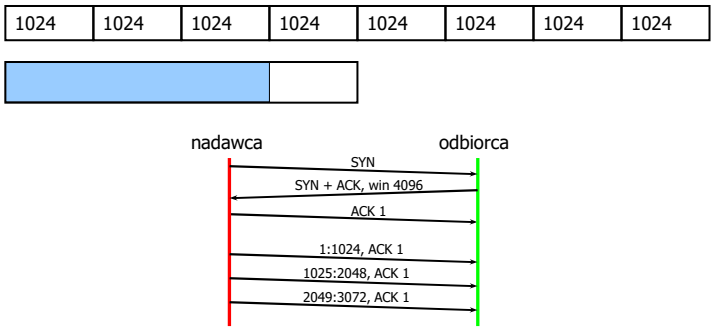
Przesuwane okno



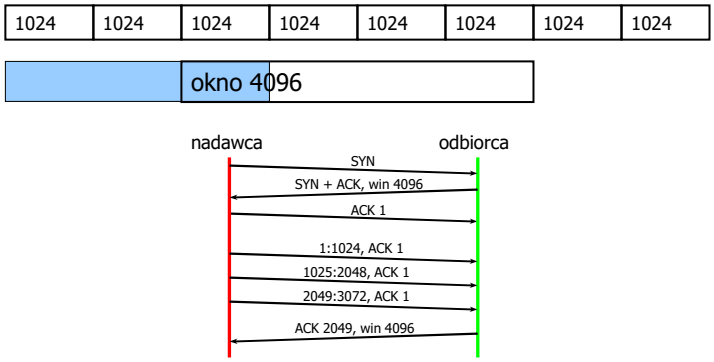
Przesuwane okno



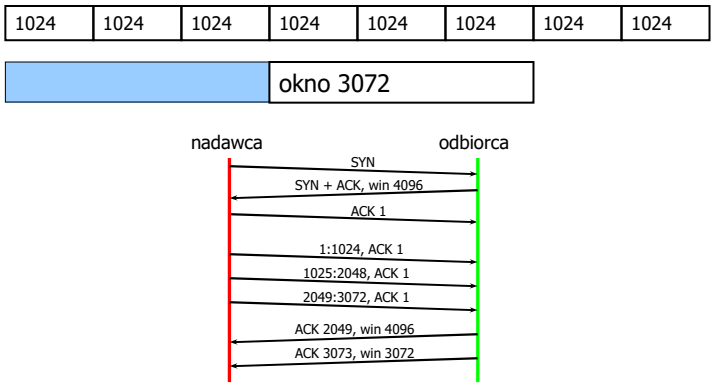
Przesuwane okno



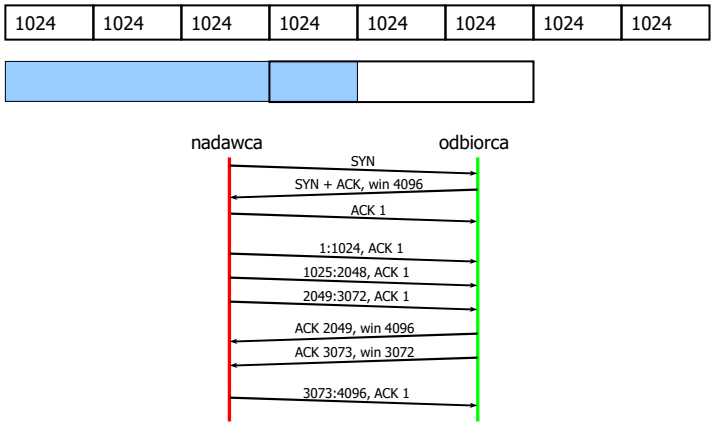
Przesuwane okno



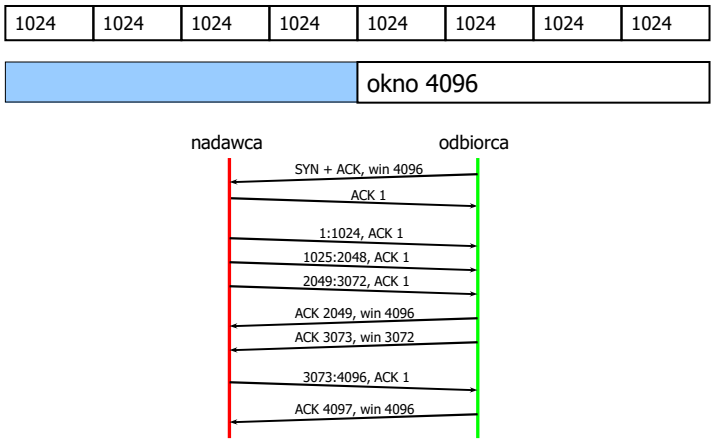
Przesuwane okno



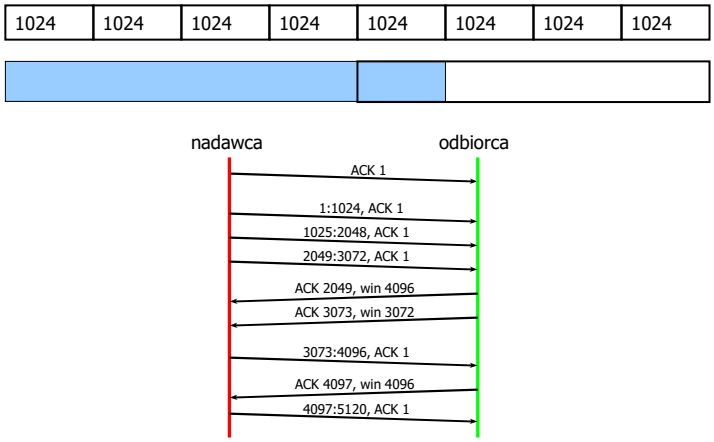
Przesuwane okno



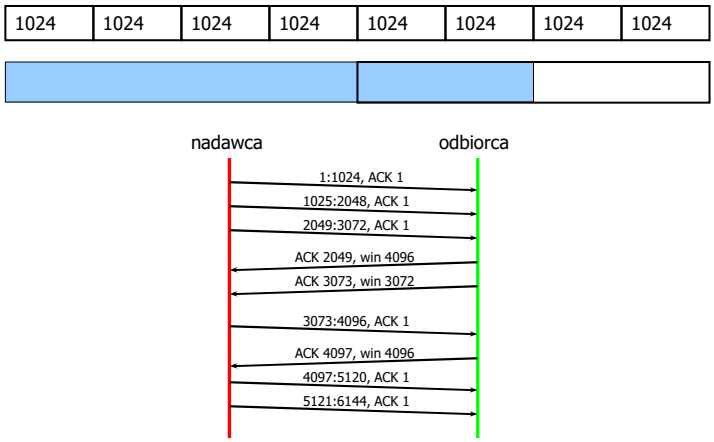
Przesuwane okno



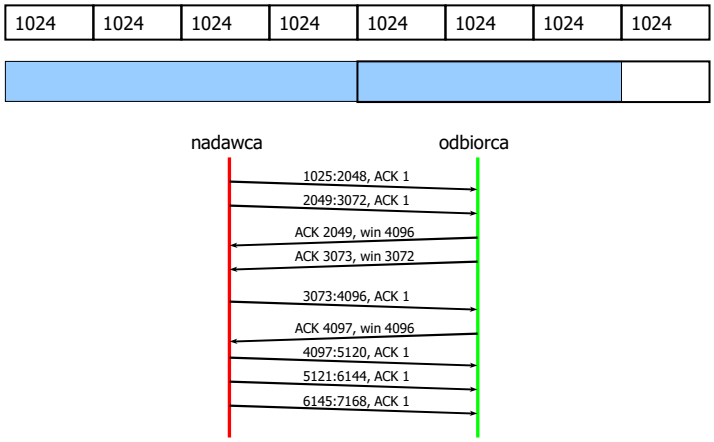
Przesuwane okno



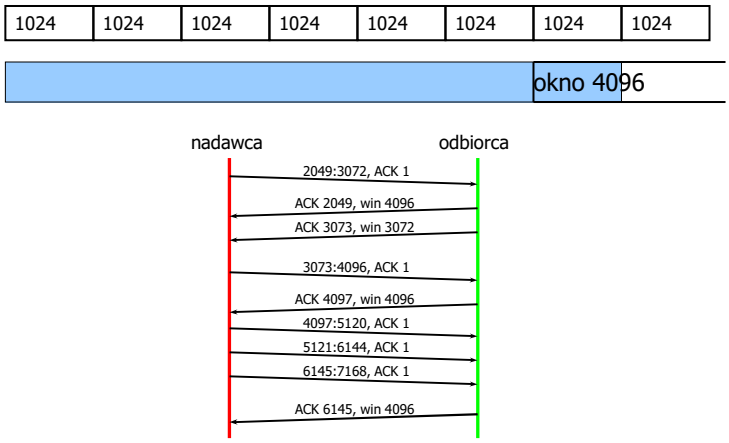
Przesuwane okno



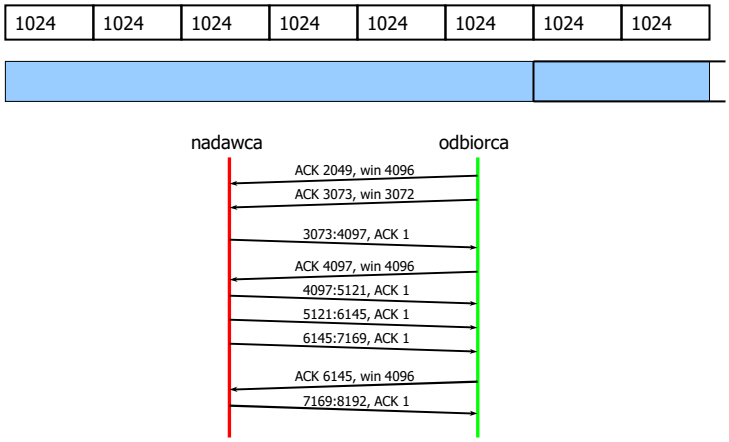
Przesuwane okno



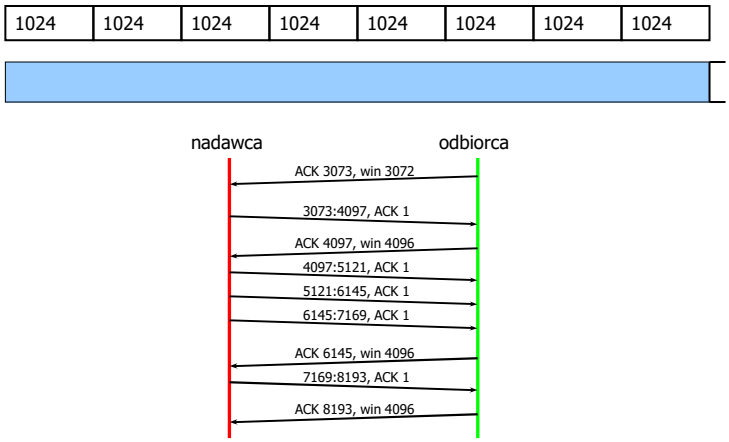
Przesuwane okno



Przesuwane okno



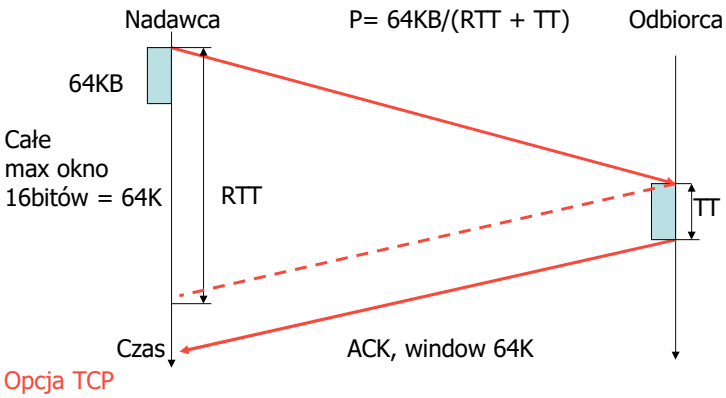
Przesuwane okno



Przesuwane okno

- Problem: okno o rozmiarze 0
  - możliwe jest zakleszczenie, gdy zginie ogłoszenie o niezerowym rozmiarze okna
- Rozwiązanie: nadawca co jakiś czas „próbkuje” zerowe okno
- Dane pilne (z ustawioną flagą URG) są wysyłane mimo zerowego rozmiaru okna
  - tego typu dane są niejako „poza” głównym strumieniem transmisyjnym

Max. długość okna a przepustowość TCP



Opcja TCP

Rozwiązanie: Skalowanie okna = mnożnik 16 bitowy = max okno 2^{32}

Powolny start

- Natychmiastowe zapełnienie całego okna może prowadzić do powstawania zatorów
- Nadawca określa okno przeciążenia *cwnd* (ang. congestion window) tj. ilość segmentów, które będzie wysyłał
  - Powolny start — kontrola przepływu przez nadawcę
  - Wielkość okna — kontrola przepływu przez odbiorcę
- Początkowo cwnd = 1
- Zwiększanie cwnd o 1 po otrzymaniu ACK

Powolny start

- Przy stwierdzeniu przeciążenia (przekroczenie czasu oczekiwania na potwierdzenie) cwnd jest zmniejszane dwukrotnie (o ile cwnd > 1 segment)
  - przy powtarzających się stratach cwnd maleje wykładniczo
- Następnie znowu każdy potwierdzony segment powoduje zwiększanie cwnd o 1 aż do momentu, gdy cwnd przyjmie wartość równą połowie wartości, przy której wystąpiło przeciążenie. Wtedy cwnd zaczyna rosnąć wolniej (jest zwiększane o 1 tylko wtedy, gdy wszystkie segmenty w oknie zostały potwierdzone).

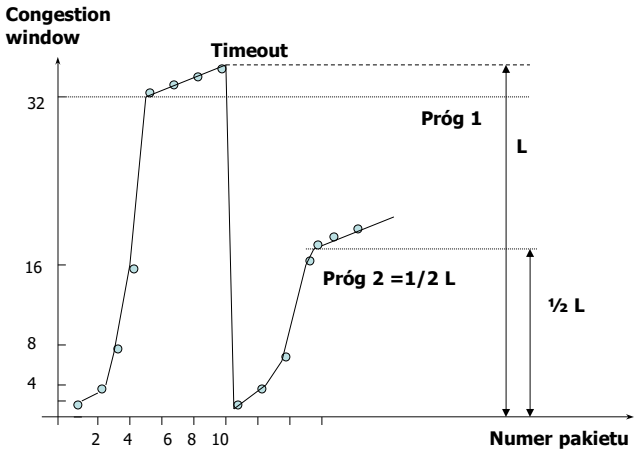
Powolny start

- Liczba wysyłanych bajtów =

  Minimum { Sending window,
  Congestion window }

- Powolny start jest uaktywniany po każdym **Timeout**

Powolny start



Przesyłanie danych interaktywnych

- Opóźnione potwierdzenie
- Algorytm Nagle'a

Dla efektywności protokołu TCP kluczowe znaczenie ma możliwość wysyłania dużych bloków danych

Kompromis pomiędzy czasem reakcji a efektywnością

Kto jest odpowiedzialny za wielkość bloków danych ?

Nadawca: wysyła znak po znaku

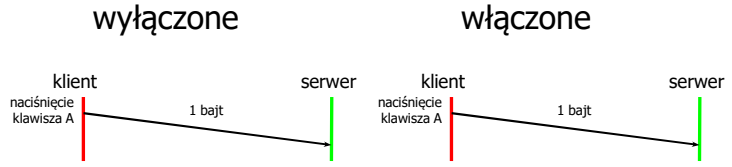
- Opóźnione potwierdzenie
- Algorytm Nagle'a

Odbiorca: odbiera znak po znaku

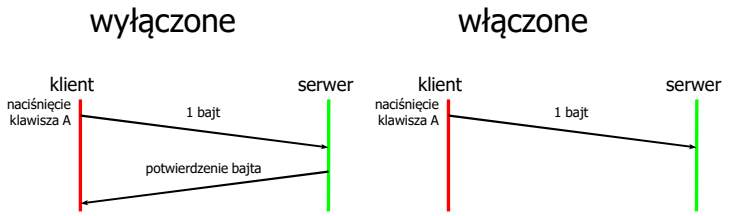
- Syndrom głupiego okna (Silly window syndrome)

Opóźnione potwierdzenie

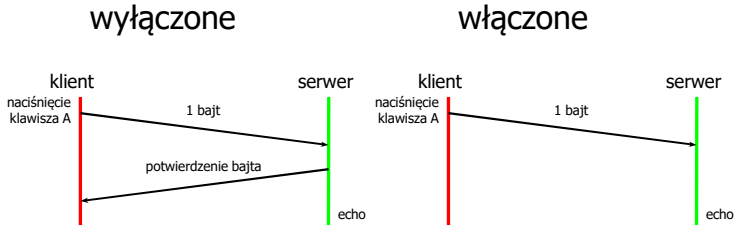
Opóźnione potwierdzenie



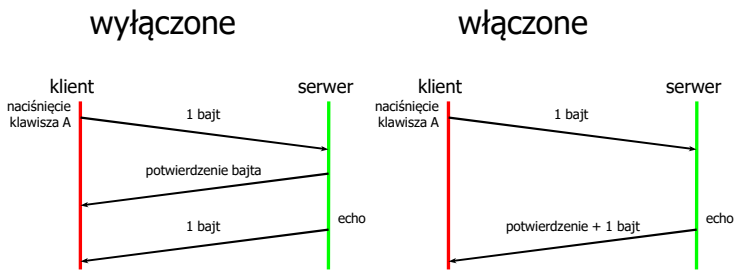
Opóźnione potwierdzenie



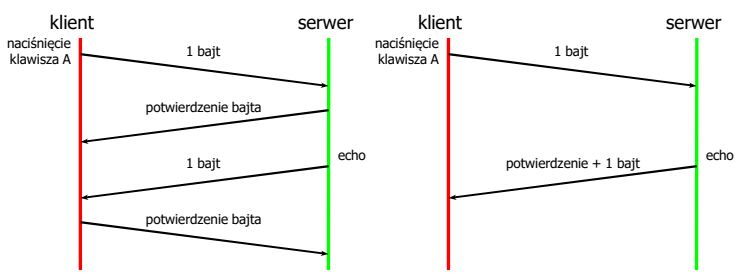
Opóźnione potwierdzenie



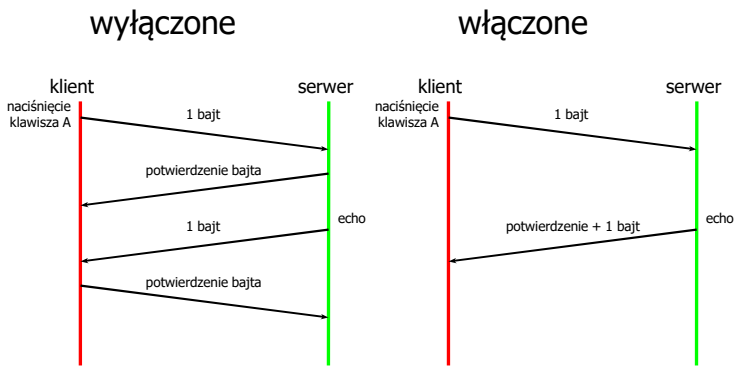
Opóźnione potwierdzenie



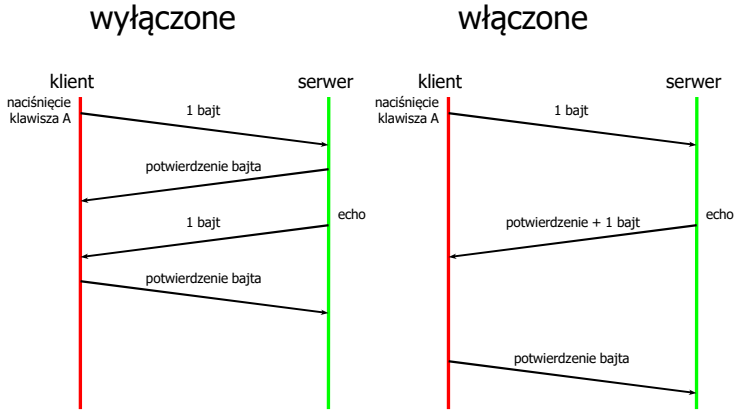
Opóźnione potwierdzenie



Opóźnione potwierdzenie



Opóźnione potwierdzenie



Algorytm Nagle'a

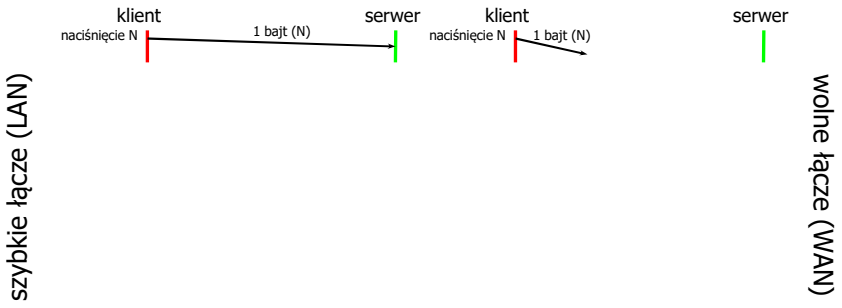
- Zastosowanie: Nadawca wysyła dane w postaci pojedynczych bajtów (małych bloków).
- Algorytm:

```
wyślij pierwszy bajt,
oczekuj na potwierdzenie,
wyślij zebrane bajty,
```

- Samoregulacja
- Oszczędność łącza, szczególnie dla wolnych sieci rozległych
- Potrzeba wyłączenia algorytmu Nagle'a dla danych interaktywnych (np. zdalny pulpit/ruch myszką)

Algorytm Nagle'a

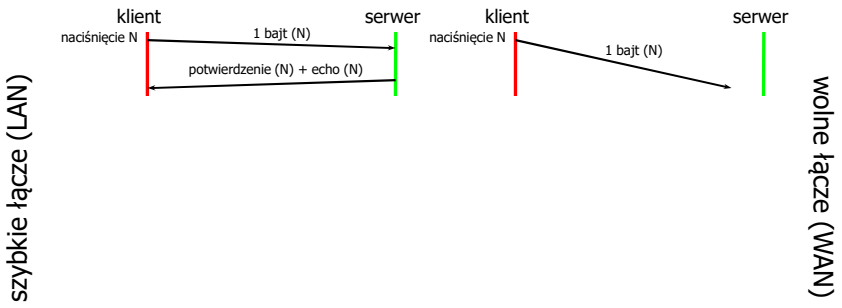
Algorytm Nagle'a




```
klient:~> rlogin serwer
serwer:~> _
```

```
klient:~> rlogin serwer
serwer:~> _
```

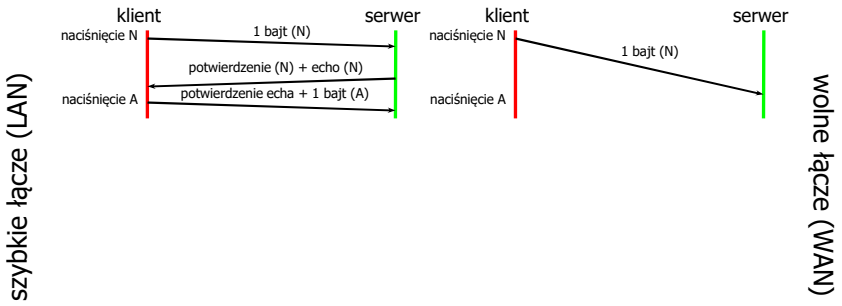
Algorytm Nagle'a




```
klient:~> rlogin serwer
serwer:~> n_
```

```
klient:~> rlogin serwer
serwer:~> _
```

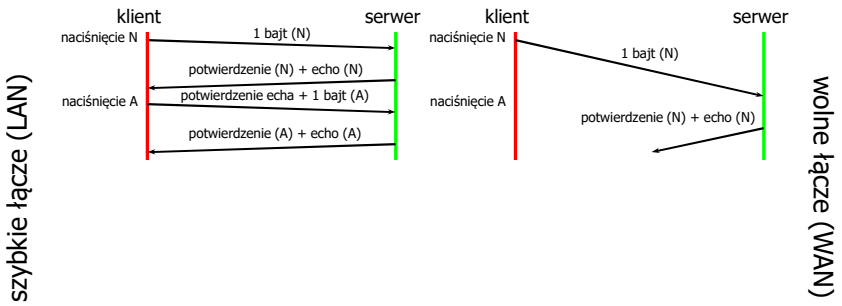
Algorytm Nagle'a




```
klient:~> rlogin serwer
serwer:~> n_
```

```
klient:~> rlogin serwer
serwer:~> _
```

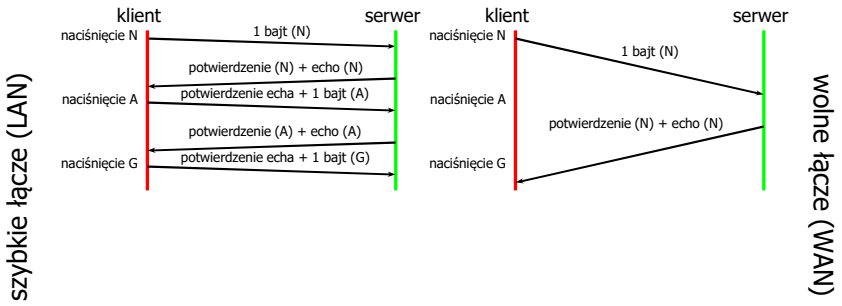
Algorytm Nagle'a




```
klient:~> rlogin serwer
serwer:~> na_
```

```
klient:~> rlogin serwer
serwer:~> _
```

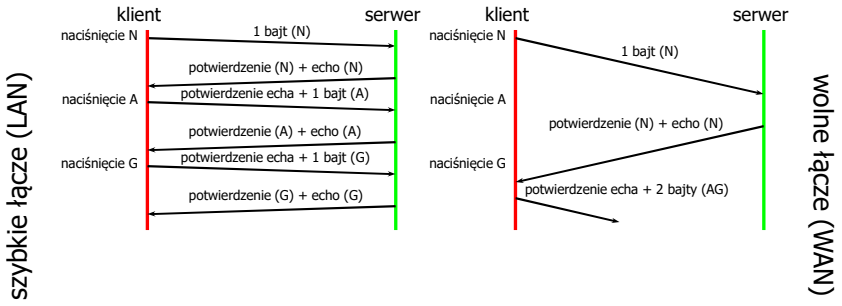
Algorytm Nagle'a




```
klient:~> rlogin serwer
serwer:~> na_
```

```
klient:~> rlogin serwer
serwer:~> n_
```

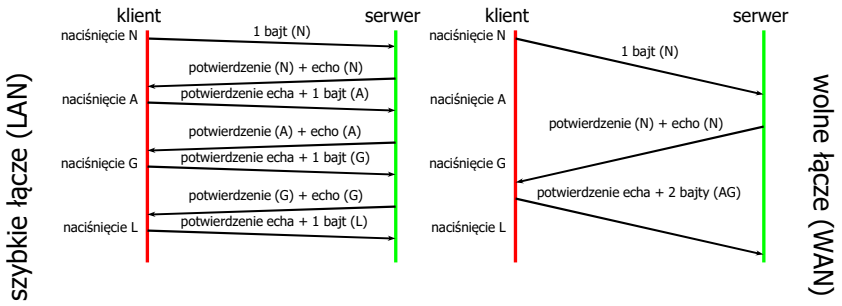
Algorytm Nagle'a




```
klient:~> rlogin serwer
serwer:~> nag_
```

```
klient:~> rlogin serwer
serwer:~> n_
```

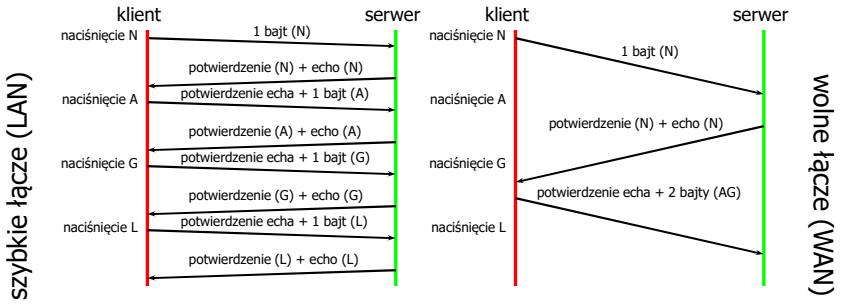
Algorytm Nagle'a




```
klient:~> rlogin serwer
serwer:~> nag_
```

```
klient:~> rlogin serwer
serwer:~> n_
```

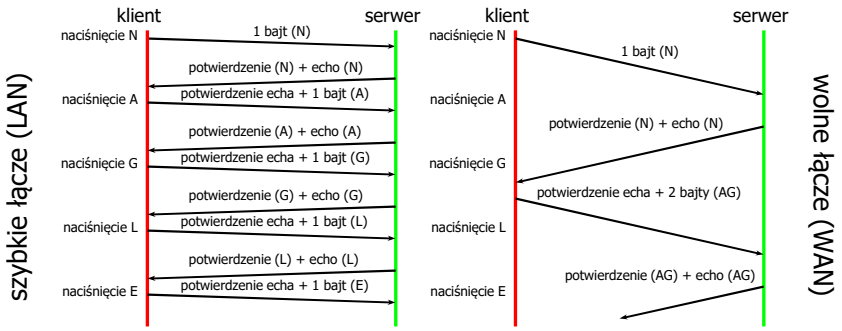
Algorytm Nagle'a




```
klient:~> rlogin serwer
serwer:~> nagl_
```

```
klient:~> rlogin serwer
serwer:~> n_
```

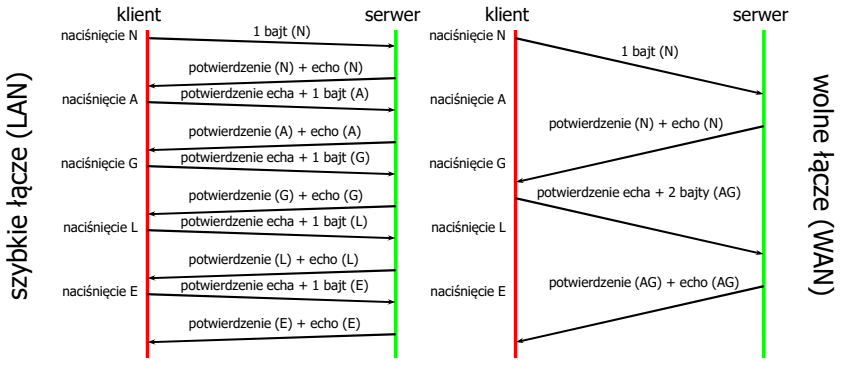
Algorytm Nagle'a




```
klient:~> rlogin serwer
serwer:~> nagl_
```

```
klient:~> rlogin serwer
serwer:~> n_
```

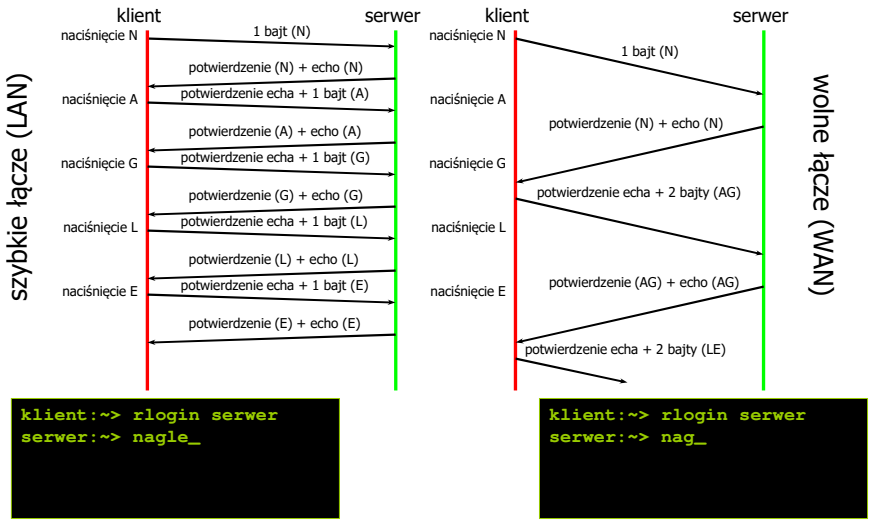
Algorytm Nagle'a




```
klient:~> rlogin serwer
serwer:~> nagle_
```

```
klient:~> rlogin serwer
serwer:~> nag_
```

Algorytm Nagle'a



Syndrom głupiego okna

Problem:

Odbiorca odczytuje bufor bajt po bajcie (małymi blokami)

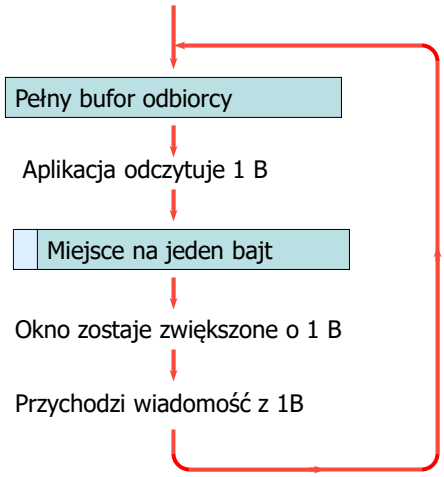
⇩

Odbiorca ogłasza niewielkie okna zamiast poczekać na opróżnienie bufora

⇩

Nadawca wysyła niewielkie segmenty, zamiast większych bloków danych

Syndrom głupiego okna



Syndrom głupiego okna

Rozwiązanie:

Odbiorca czeka z otwarciem okna, aż możliwe będzie powiększenie go o MSS (Maximum Segment Size) lub o połowę rozmiaru bufora odbiorcy

Rozwiązanie Clark'a (Clark's solution)

Clark + Nagle

- Nadawca transmituje dane, gdy może wysłać:
  - cały segment danych MSS
  - co najmniej połowę rozmiaru bufora odbiorcy
  - dowolną ilość danych pod warunkiem, że potwierdzone zostały wszystkie wysłane segmenty (przy włączonym alg. Nagle'a)

Nadawca nie wysyła małych segmentów (Nagle), a odbiorca ich nie żąda (Clark)

Zegary TCP

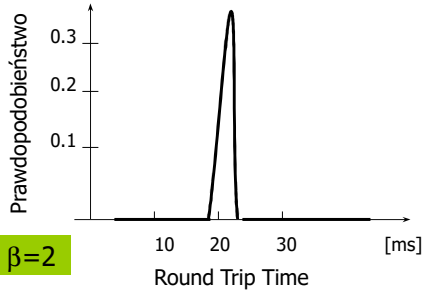
Retransmission Timer	zegar odmierzający Timeout; czas oczekiwania jest zmienny (retransmisja z adaptacją)
Persistence Timer	zapobiega zakleszczeniu w sytuacji zgubienia pakietu zwiększającego okno
Keepalive Timer	pozwala na sprawdzenie aktywności drugiej strony połączenia

Retransmission Timer

$RTT = \alpha \times RTT_{stare} + (1-\alpha) \times M$

M – czas otrzymania potwierdzenia („nowe" RTT)

$\alpha = 7/8$



Sposób 1

Timeout = $\beta \times$ RTT, gdzie zalecane β=2

Sposób 2 (alg. Jacobsona)

$D = \alpha \times D + (1- \alpha) \times |RTT - M|$

Timeout = $RTT + 4 \times D$

Retransmission Timer cd.

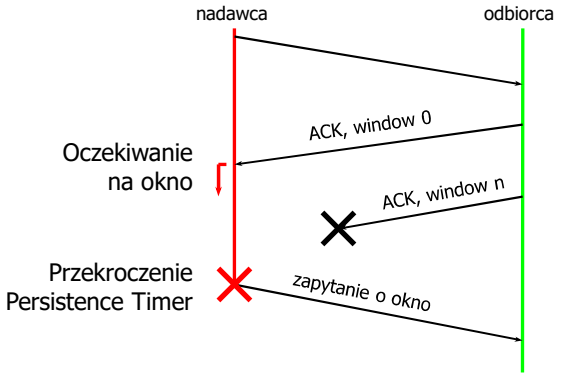
Dynamiczna estymacja RTT jest przeprowadzana wg. Algorytmu Karna

W sytuacji braku potwierdzenia nie jest zmieniane RTT lecz zwiększa się Timeout 2 dwa razy (zazwyczaj).

Ponowne wysłanie może spowodować niejednoznaczność, które ACK odpowiada ostatniemu wysłaniu.

Dlatego nie zmienia się RTT na podstawie czasów zmierzonych dla retransmitowanych segmentów.

Persistence Timer



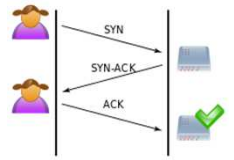
Zastosowania (TCP)

- Przesyłanie danych wrażliwych na gubienie pakietów
  - dane interaktywne: Telnet, SSH, mysz w X-Windows itp.
  - dane masowe: FTP, Mail, News itp.
- Nie pozwala na transmisję grupową (multicasting) — dokładnie dwa końce połączenia

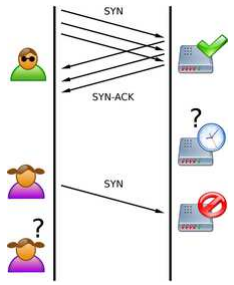
Atak DoS

- DoS (Denial of Service) – odmowa obsługi/dostępu do usługi
- SYN Flood
  - popularny typ ataków DoS z wykorzystaniem mechanizmu nawiązania połączenia TCP (Three way handshake)
  - wykorzystanie zainfekowanych komputerów jako zombie które wysyłają zapytania.

Grafika: pl.wikipedia.org/wiki/SYN_flood



Normalne nawiązanie połączenia



SYN Flood

Atak DoS

- Procedura ataku SYN Flood:
  - aktywacja komputerów zombie które wysyłają wiele zapytań SYN,
  - komputer atakowany odpowiada SYN-ACK, alokując odpowiednie zasoby,
  - nie dochodzi do połączenia (komputer oczekuje na ACK od zombie),
  - wiele fałszywych połączeń blokuje ofiarę i uniemożliwia połączenia od innych użytkowników (brak odpowiedzi na próby nawiązania połączenia SYN).

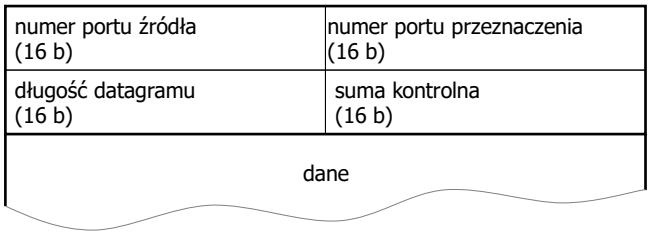
User Datagram Protocol

- Bardzo prosty bezpołączeniowy protokół warstwy transportowej
  - szybki (małe narzuty)
  - zawodny
    - nie zawiera żadnych komunikatów potwierdzających przyjęcie pakietu bądź informujących o jego zagubieniu,
    - nie gwarantuje, że dane zostaną dostarczone do procesu docelowego w kolejności wysłania,
    - pakiety danych mogą być zduplikowane,
    - nie zawiera żadnego mechanizmu kontroli prędkości przesyłania danych pomiędzy hostami.

Proces docelowy musi sobie „radzić”
ze wszystkimi wymienionymi zjawiskami.

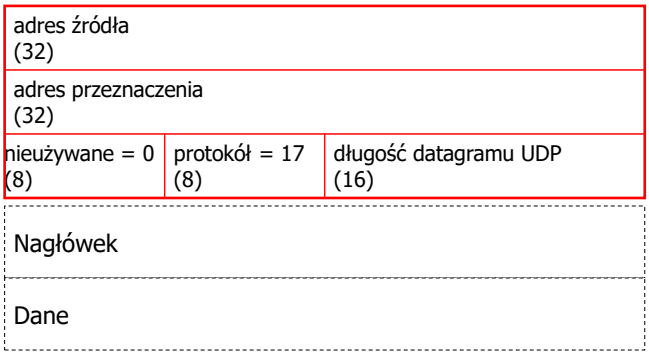
Nagłówek UDP

- Nagłówek ma rozmiar 8 bajtów
- Suma kontrolna jest nieobowiązkowa (0 = brak sumy)
  - gdyby suma po wyliczeniu wyniosła 0, zapisuje się ją jako 0xFFFF (stosowany jest kod U1)



Suma kontrolna

- Obliczanie sumy kontrolnej dla nagłówka, danych oraz pseudonagłówka



Zastosowania (UDP)

- Transmisje grupowe
- Transmisje w czasie rzeczywistym
- Przesyłanie danych mniej wrażliwych na gubienie pakietów
- Przesyłanie w sieci LAN
  - NFS, DNS

Uwaga: Częstym błędem programistów jest testowanie programów wykorzystujących UDP tylko w sieci lokalnej (o niskim stopniu zawodności). Programy takie uruchomione w sieci rozległej mogą zachowywać się zupełnie inaczej!

Stream Control Transmission Protocol

- Protokół połączeniowy sterowania transmisją strumieniową (Stream Control Transmission Protocol - SCTP): RFC 2960 – 2000
  - UDP: RFC 768 – 1980
  - TCP: RFC 793 – 1981
- Opracowany m.in. przez:
  - Motorola
  - Cisco
  - Siemens
  - Ericsson

Potrzeba

- Protokół UDP wykorzystywany w nietypowych zastosowaniach
- Obecnie TCP obsługuje zdecydowaną większość aplikacji
- Dla coraz większej liczby aplikacji TCP okazuje się niewystarczające np.:
  - wymagana jest niezawodność bez gwarancji kolejności,
  - TCP nie oferuje przesyłania komunikatów,
  - TCP nie wspiera hostów z wieloma interfejsami sieciowymi tzw. *multihomed*,
  - protokół TCP jest podatny na ataki typu DoS.

Charakterystyka

- Protokół połączeniowy
- Niezawodny transport danych z selektywnymi potwierdzeniami
- Przesyłanie komunikatów / także bez kontroli kolejności odbioru
- Wbudowane mechanizmy kontroli i zapobiegania przeciążeniom
- Wykrywanie MTU ścieżki i fragmentacja
- Wsparcie dla hostów z wieloma interfejsami
- Wsparcie dla komunikacji wielu strumieni w jednym połączeniu
- i inne

SCTP a TCP

Podstawowe różnice w stosunku do TCP

- TCP nie jest multihoming,
- TCP nie dopuszcza wielu strumieni w jednym połączeniu (**association**),
- TCP zakłada, że strumień to sekwencja bajtów, natomiast w SCTP strumień to sekwencja wiadomości (krótkich lub długich).

Asocjacja

- SCTP operuje na poszerzonym pojęciu asocjacji

TCP i UDP

(protokół, adres lokalny, proces lokalny, adres obcy, proces obcy)

SCTP

(protokół, adresy lokalne, port lokalny, adresy obce, port obcy)

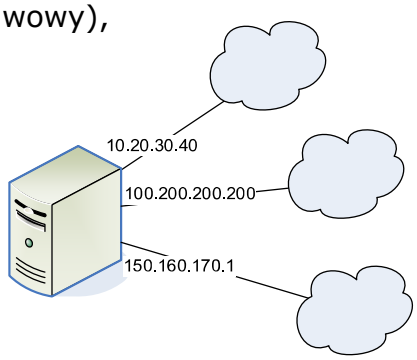
Gniazdo

- W SCTP istnieje możliwość wykorzystania wielu połączeń sieciowych (Multihoming)
- Gniazdo SCTP obejmuje:
  - jeden lub więcej adresów IP (jeden z adresów jest wykorzystywany jak podstawowy),
  - numer portu.
- Gniazdo może mieć postać:

(SCTP, [10.20.30.40, 150.160.170.1], 80)

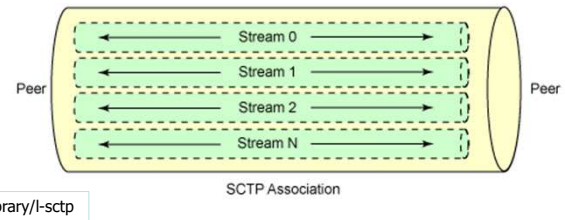
lub np.:

(SCTP, [100.200.200.200], 80)



Multistreaming

- Możliwość realizacji wielu niezależnych transmisji w ramach jednej asocjacji SCTP:
  - Przykładem jest sterowanie i transmisja danych dla protokołu FTP.
- Możliwość implementacji różnych scenariuszy:
  - pełna niezawodność,
  - częściowa niezawodność.



Grafika: www.ibm.com/developerworks/library/l-sctp

Pakiet

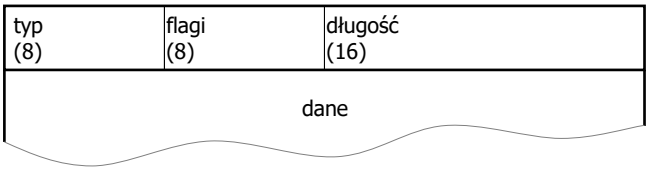
- Pakiet składa się ze:
  - wspólnego nagłówka
  - jednej lub więcej części opatrzonych własnym nagłówkiem (enkapsulacja)

Wspólny nagłówek

numer portu źródła (16)	numer portu przeznaczenia (16)
znacznik weryfikacyjny (32)	
suma kontrolna CRC-32c (32)	

- Znacznik weryfikacyjny pozwala rozróżnić pakiety z dwóch różnych połączeń

Nagłówek części



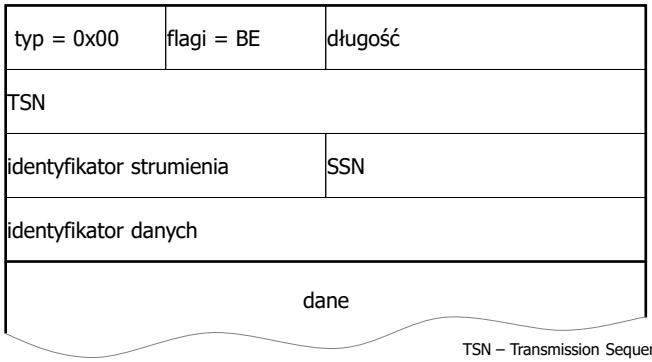
- Typy określone w standardzie obejmują m.in.:
  - DATA [0x00],
  - INIT [0x01], INIT-ACK [0x02],
  - SHUTDOWN [0x07], SHUTDOWN-ACK [0x08], SHUTDOWN-COMPLETE [0x08]
- Podział na części ułatwia rozszerzanie protokołu

Typ (Value Chunk Type)

0 - Payload Data (DATA)
1 - Initiation (INIT)
2 - Initiation Acknowledgement (INIT ACK)
3 - Selective Acknowledgement (SACK)
4 - Heartbeat Request (HEARTBEAT)
5 - Heartbeat Acknowledgement (HEARTBEAT ACK)
6 - Abort (ABORT)
7 - Shutdown (SHUTDOWN)
8 - Shutdown Acknowledgement (SHUTDOWN ACK)
9 - Operation Error (ERROR)
10 - State Cookie (COOKIE ECHO)
11 - Cookie Acknowledgement (COOKIE ACK)
12 - Reserved for Explicit Congestion Notification Echo (ECNE)
13 - Reserved for Congestion Window Reduced (CWR)
14 - Shutdown Complete (SHUTDOWN COMPLETE)
15 to 62 - reserved by IETF
63 - IETF-defined Chunk Extensions
64 to 126 - reserved by IETF
127 - IETF-defined Chunk Extensions
128 to 190 - reserved by IETF
191 - IETF-defined Chunk Extensions
192 to 254 - reserved by IETF
255 - IETF-defined Chunk Extensions

DATA [0x00]

- Ta część zawsze znajduje się na końcu pakietu



TSN – Transmission Sequence Number
SSN – Stream Sequence Number

Flagi

B	E	Description
1	0	First piece of a fragmented user message
0	0	Middle piece of a fragmented user message
0	1	Last piece of a fragmented user message
1	1	Unfragmented Message

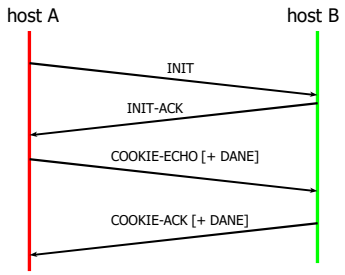
INIT [0x01]

typ = 0x01	flagi = 0	długość
początkowy znacznik weryfikacyjny		
okno odbiorcy		
liczba strumieni wyjściowych		max. liczba strumieni wejściowych
początkowy TSN		
parametry opcjonalne (adresy IP asocjacji)		

Rozszerzenia protokołu

- Podział na pakietu na wspólny nagłówek i jedną lub więcej części ułatwia rozszerzanie protokołu
- Co z obsługą nowych rozszerzeń?
- Obsługa zależy od pierwszych dwóch bitów typu:
  - 00xx xxxx – jeśli typ jest nieznany to porzuć pakiet „po cichu” i przerwij dalszą analizę,
  - 01xx xxxx – jeśli typ jest nieznany to porzuć pakiet, zgłoś błąd do nadawcy i przerwij dalszą analizę pakietu,
  - 10xx xxxx – jeśli typ jest nieznany to porzuć pakiet „po cichu”, ale analizuj kolejne części,
  - 11xx xxxx – jeśli typ jest nieznany to porzuć pakiet, zgłoś błąd do nadawcy, ale analizuj kolejne części.

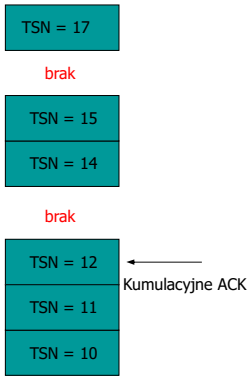
Przykład nawiązania połączenia



SACK (Selective Acknowledgment)

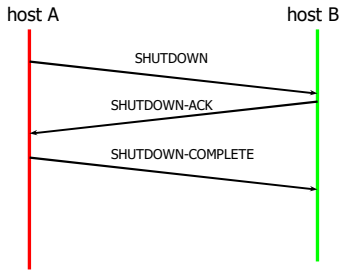
typ = 0x03	flagi	długość
kumulacyjne TSN ACK		
okno odbiorcy		
liczba opuszczonych bloków N		liczba zdublowanych bloków M
Początek opuszczonego bloku #1		Koniec opuszczonego bloku #1
. . .		
TSN zdublowanego bloku #1		
. . .		

SACK - przykład



12	
4660	
2	0
2	3
5	5

Zamknięcie połączenia



Podsumowanie (SCTP)

- Służy do przesyłania strumienia wiadomości (w przeciwieństwie do TCP gdzie jest przesyłany strumień bajtów)
- Zapewnia działanie takich samych usług jak TCP (transmisja niezawodna, możliwość zagwarantowania kolejności, retransmisje, brak przeciążeń)
- Multihoming:
  - transmisja przez wiele łączy,
  - zakończenia połączeń mogą mieć wiele adresów IP.
- Zaprojektowany dla usług które mają uwarunkowania czasowe np. telefonii IP (VoIP)
- Umożliwia zwiększenie wydajności i niezawodności przy transmisji z wykorzystaniem protokołu IP

Porównanie - TCP/UDP/SCTP

	TCP	UDP	SCTP
Tryb	połączeniowy	bezpołączeniowy	połączeniowy
Transmisja	strumień bajtów	wiadomości (datagramy)	wiadomości (datagramy)
Sterowanie przepływem	tak	nie	tak
Uporządkowanie	ścisłe	brak	częściowe
Niezawodność	retransmisje	brak	retransmisje

Inne protokoły warstwy transportu

Datagram Congestion Control Protocol

- Datagram Congestion Control Protocol (DCCP) – opracowany w 2002 (standaryzacja 2006) - RFC 4340:
  - zaprojektowany do transmisji interaktywnych bez wymogu niezawodności (strumieniowanie, gry wieloosobowe, VoIP),
  - brak obsługi transmisji grupowej (multicast),
  - połączeniowy,
  - kontrola przeciążeń,
  - transmisja zawodna (niezawodne zestawienie i zakończenie połączenia),
  - bez kontroli przepływu.

Resource Reservation Protocol

- Resource Reservation Protocol (RSVP) - RFC2205:
  - konfiguracja zasobów w systemach Integrated Services (QoS),
  - działa w oparciu o IPv4 lub IPv6,
  - nie jest protokołem transmisji danych ani routingu,
  - umożliwia zgłoszenie wymagań dotyczących przepustowości łącza i na ich podstawie dokonuje rezerwacji zasobów wzdłuż trasy przesyłania danych,
  - wykorzystywany dla transmisji strumieniowych audio/wideo.

Literatura

- W.R. Stevens *Biblia TCP/IP*, t. 1, 2
- W.R. Stevens *Programowanie zastosowań sieciowych w systemie UNIX*
- D. Comer *TCP/IP* t.2
- www.iana.org/assignments/port-numbers lub RFC 1700
- RFC 2001, 2018, 2581, 2582, 2960, 3042

KONIEC