SIECI KOMPUTEROWE

wykład dla kierunku informatyka

semestr 4 i 5

dr inż. Michał Sajkowski

Instytut Informatyki PP

pok. 227G PON PAN, Wieniawskiego 17/19

Michal.Sajkowski@cs.put.poznan.pl

tel. +48 (61) 8 582 100

http://www.man.poznan.pl/~michal/

sieci komputerowe
wykład 5
protokoły końcowe

literatura podstawowa

wykład prawie w całości przygotowany na podstawie tekstu i rysunków z rozdziału 6 w książce:

L.L. Peterson, B.S. Davie
„Sieci komputerowe. Podejście systemowe”
Wydawnictwo Nakom, Poznań 2000

problemy

- *dotychczas analizowaliśmy techniki łączenia zbioru komputerów:*
- łącza bezpośrednie (włącznie z sieciami lokalnymi, takimi jak Ethernet i FDDI)
- sieci z komutacją pakietów (włącznie z ATM)
- intersieci
- *kolejnym problemem jest* przeniesienie dostawy pakietów między komputerami na poziom kanału komunikacji między procesami
- taką rolę pełni *poziom transportowy*, na którym realizowane są *protokoły końcowe*

oczekiwania aplikacji od protokołu transportowego

- gwarancja dostawy komunikatów
- dostawa komunikatów w kolejności nadawania
- dostawa co najwyżej jednej kopii komunikatu
- obsługa dowolnie dużych komunikatów
- obsługa synchronizacji między nadawcą a odbiorcą
- zezwolenie odbiorcy na nałożenie sterowania przepływem na nadawcę
- obsługa wielu procesów aplikacji na każdym komputerze

ograniczenia ze strony sieci doświadczane przez protokół transportowy

- strata komunikatów
- zmiana kolejności komunikatów
- dostawa duplikatów danego komunikatu
- ograniczenie komunikatów do pewnego skończonego rozmiaru
- dostawa komunikatów po dowolnie długim czasie

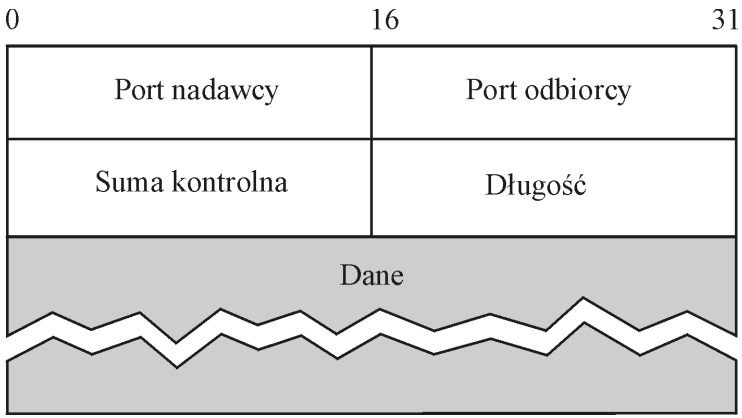
- taka sieć wyświadcza usługę *na poziomie dostępnych możliwości*, przykładem takiej sieci jest Internet

zakres wykładu

- algorytmy zamieniające niepożądane cechy sieci na wysoki poziom usług transportowych, wymagany przez programy aplikacji
- algorytmy te w kontekście usług: prostej demultiplek-sacji asynchronicznej, niezawodnego strumienia baj-tów i usługi żądanie/odpowiedź
- usługa demultipleksacji - UDP
- usługa strumienia bajtów - TCP
- algorytmy realizujące usługę żądanie/odpowiedź- RPC, SunRPC
- interfejs programowania aplikacji API
- efektywność różnych protokołów transportowych

prosty demultiplekser (UDP)

- *najprostszy protokół*: rozszerza usługę dostawy między komputerami na usługę komunikacji między procesami, dodając funkcję *demultipleksacji*, pozwalającą wielu procesom aplikacji współdzielić jedną sieć
- procesy identyfikują się wzajemnie - *pośrednio* -za pomocą portu, za pierwszym razem: *w dobrze znanym porcie* (n.p. w ustalonym porcie serwera)
- ⟨port, komputer⟩ to *klucz demultipleksacji*
- format nagłówka UDP:



porty UDP i TCP (RFC 1700)

- poniżej 255 - zastosowania publiczne

  porty TCP: FTP - port 21

  telnet - port 23

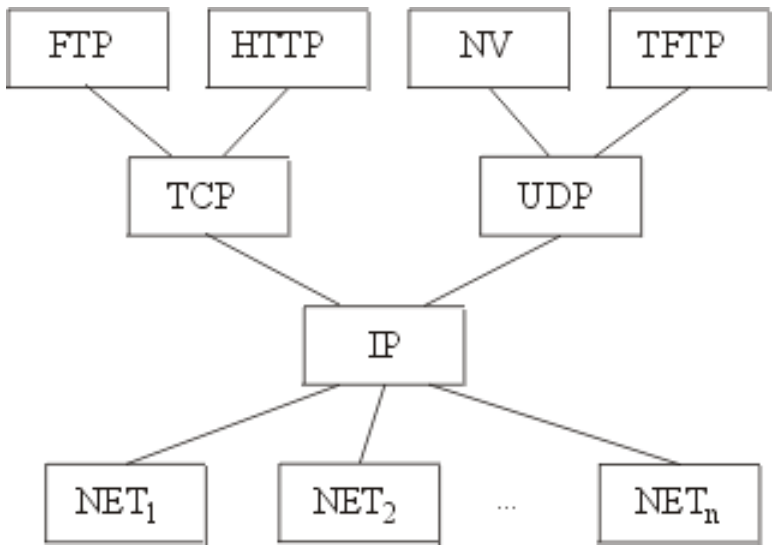
  SMTP - port 25

  HTTP - port 80

  porty UDP: DNS - port 53

  TFTP - port 69

  SNMP - port 161



- 255 - 1023 - firmy komercyjne

  talk w UNIX: port 517

- powyżej 1023 - brak uregulowań

UDP - podsumowanie

- usługa datagramowa, bezpołączeniowa, zawodna
- brak potwierdzeń, sterowania przepływem i składania pakietów
- gwarancja poprawności komunikatu za pomocą sumy kontrolnej

TCP - podsumowanie

- usługa kanałów wirtualnych, połączeniowa, niezawodna
- segmenty dostarczane w kolejności
- są potwierdzenia, jest sterowanie przepływem i składanie segmentów w komunikaty
- protokół pełnodupleksowy (jednocześnie para strumieni bajtów w obu kierunkach)
- sterowanie przepływem dla każdego strumienia
- demultipleksacja
- mechanizm kontroli przeciążenia (ochrona przed wprowadzeniem zbyt dużej ilości danych do sieci)

niezawodny strumień danych (TCP)

- *kwestie komunikacji końcowej*

  protokół przesuwnego okna (w Internecie)

  faza nawiązania połączenia

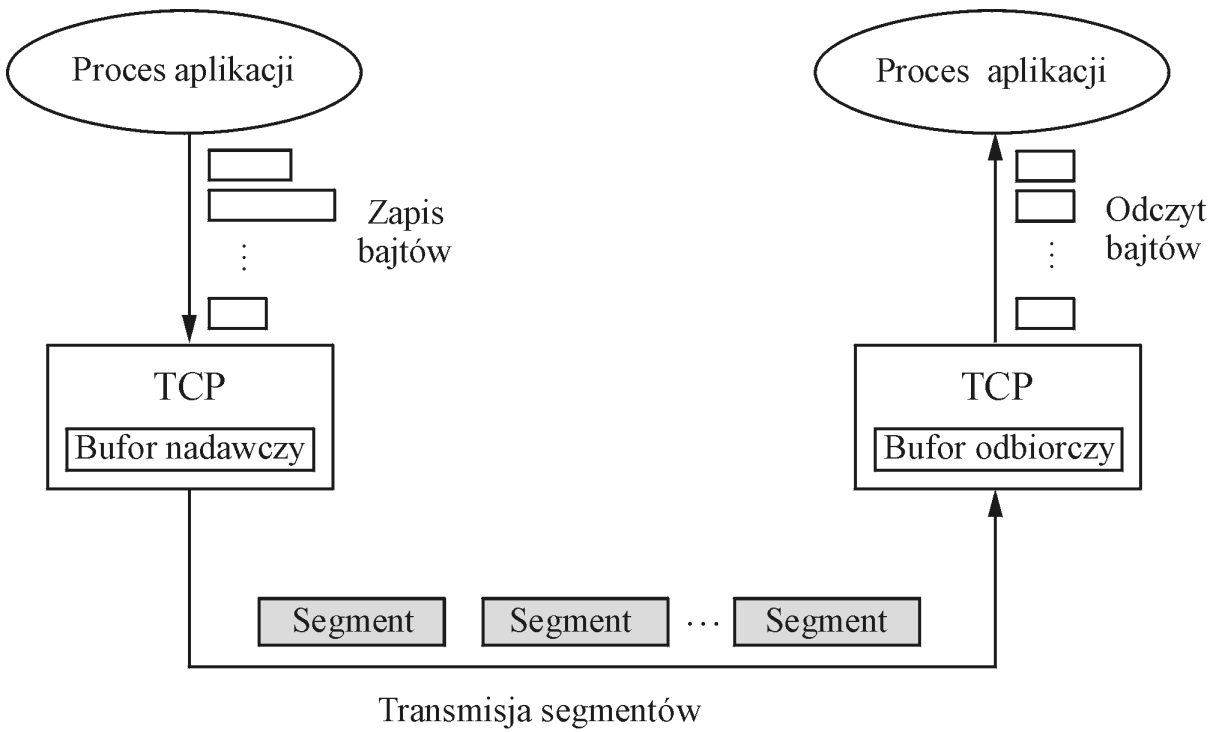
  faza wymiany danych

  faza rozłączenia połączenia

  adaptacyjny mechanizm czasu oczekiwania
- TCP jest protokołem znakowym! (nadawca nadaje bajty do połączenia TCP, a odbiorca odczytuje bajty z tego połączenia)

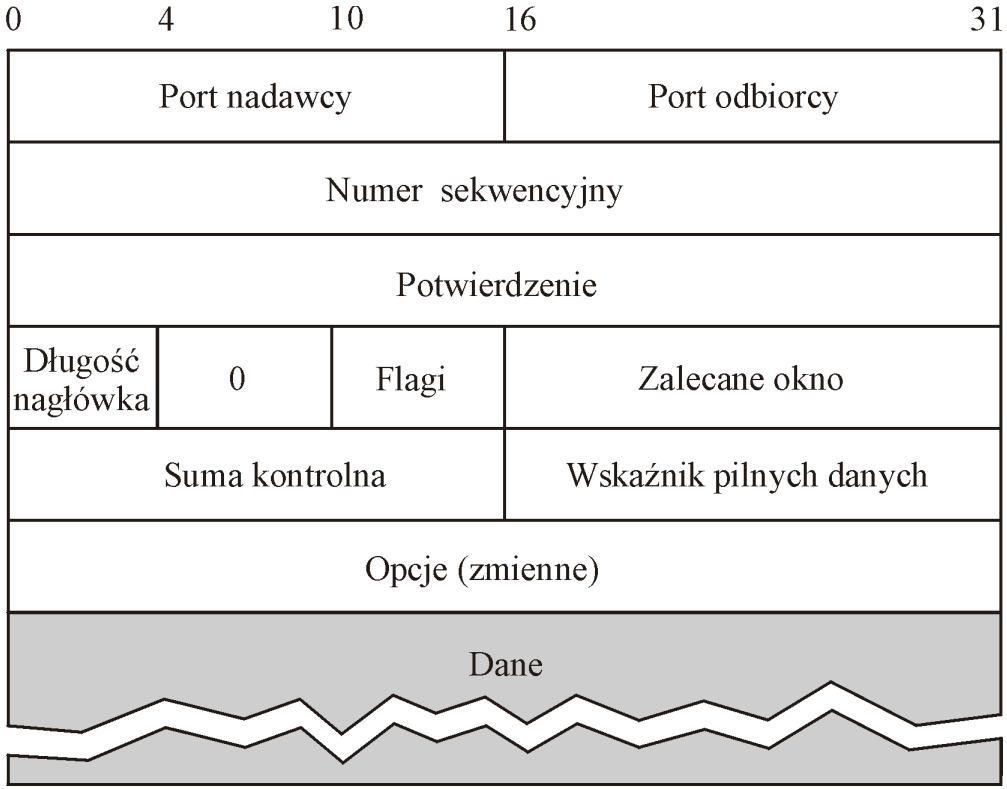
sposób zarządzania strumieniem bajtów

- *wyzwalanie transmisji segmentu*: zmienna progowa, na życzenie, za pomocą zegara



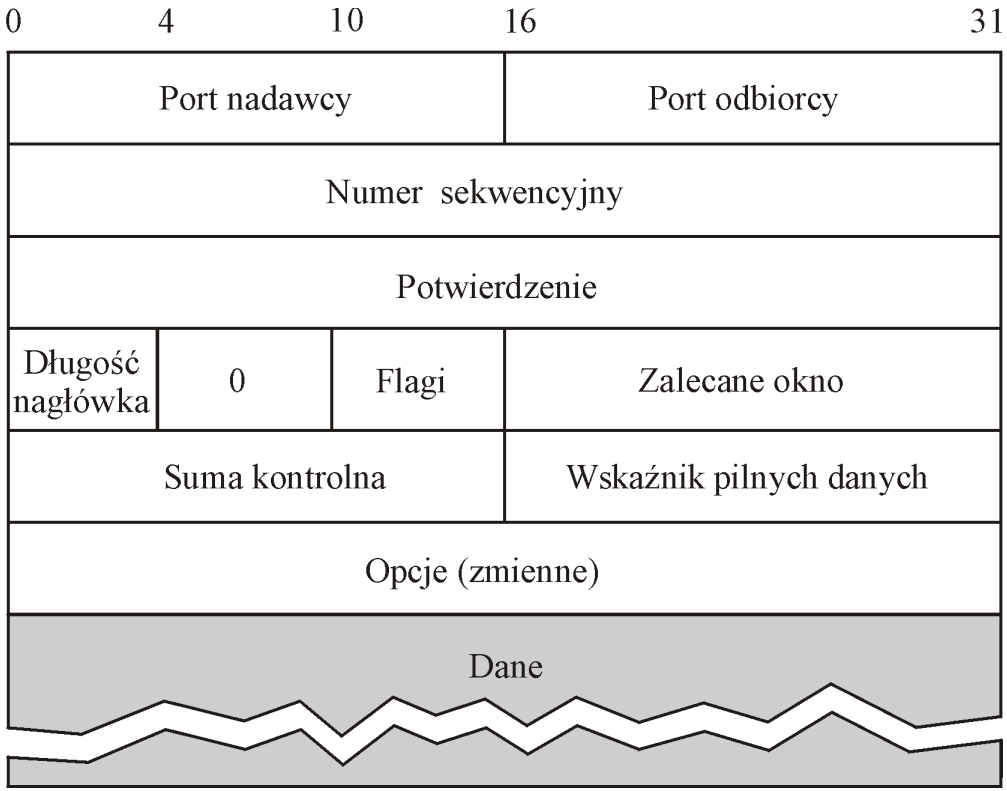
format segmentu

- porty jednocześnie z adresami IP nadawcy i odbiorcy identyfikują *połączenie* TCP (klucz demultipleksacji jest wtedy czwórką: dwa porty, dwa adresy IP)



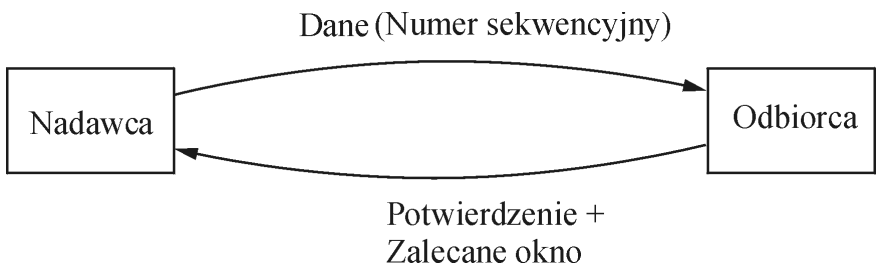
format segmentu

- *algorytm przesuwnego okna* wykorzystuje pola: numer sekwencyjny, potwierdzenie i zalecane okno



ilustracja procesu TCP

- algorytm przesuwnego okna wykorzystuje pola: numer sekwencyjny, potwierdzenie i zalecane okno

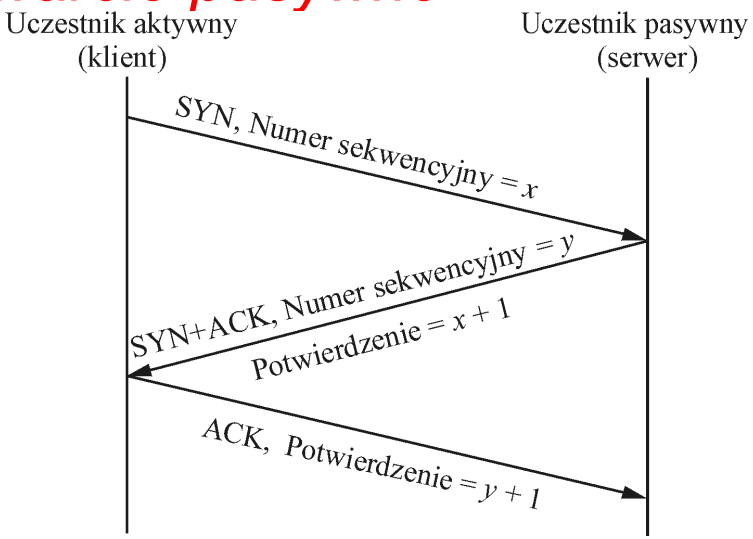


format segmentu

- *pole flagi* (6 bitów) do przenoszenia informacji sterującej między partnerami TCP:
- SYN: nawiązywanie połączenia TCP
- FIN: zamykanie połączenia TCP
- ACK: potwierdzenie jest ważne
- URG: segment zawiera dane pilne
- PUSH: wywołana operacja umieszczania na stosie
- RESET: odbiorca przerywa połączenie

nawiązanie połączenia

- *wymiana trójetapowa:* otwarcie połączenia ma charakter asynchroniczny
- *klient* (wywołujący, nawiązujący połączenie) wykonuje *otwarcie aktywne*
- *serwer* (wywoływany, akceptuje połączenie) wykonuje *otwarcie pasywne*



wymiana trójetapowa

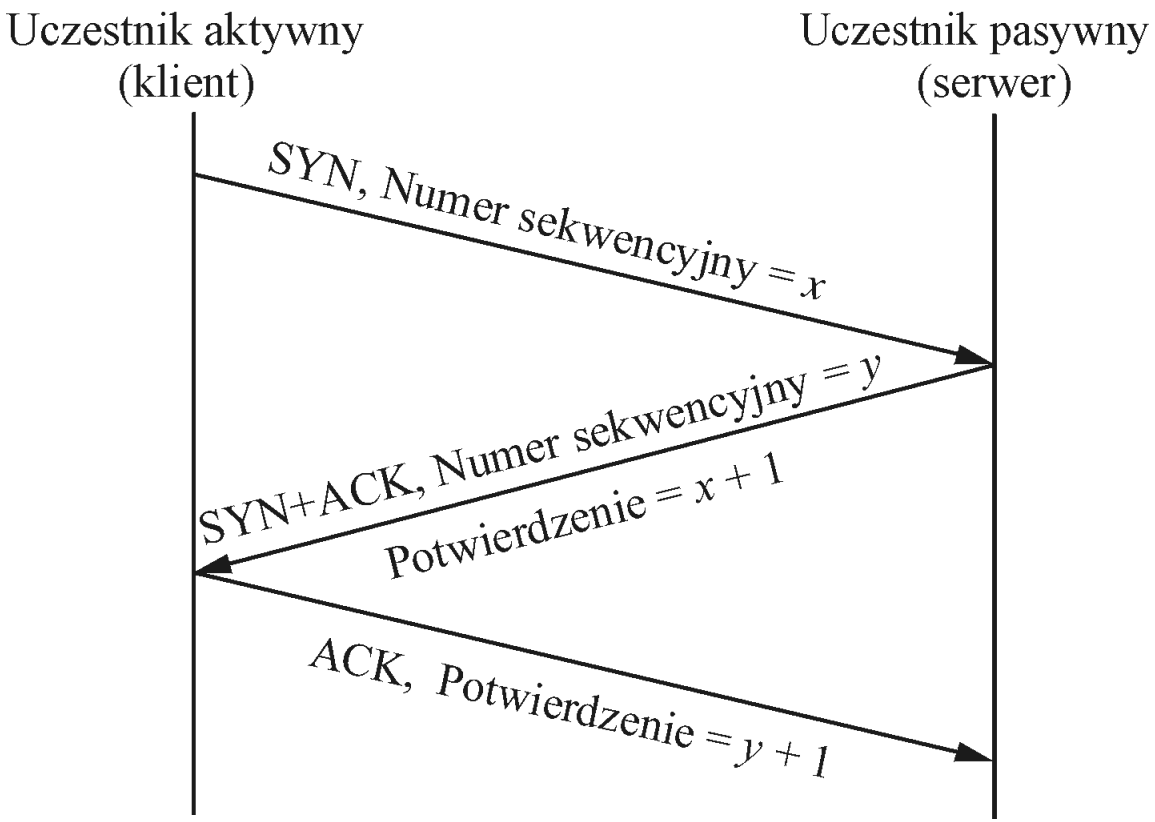


diagram przejść stanów TCP

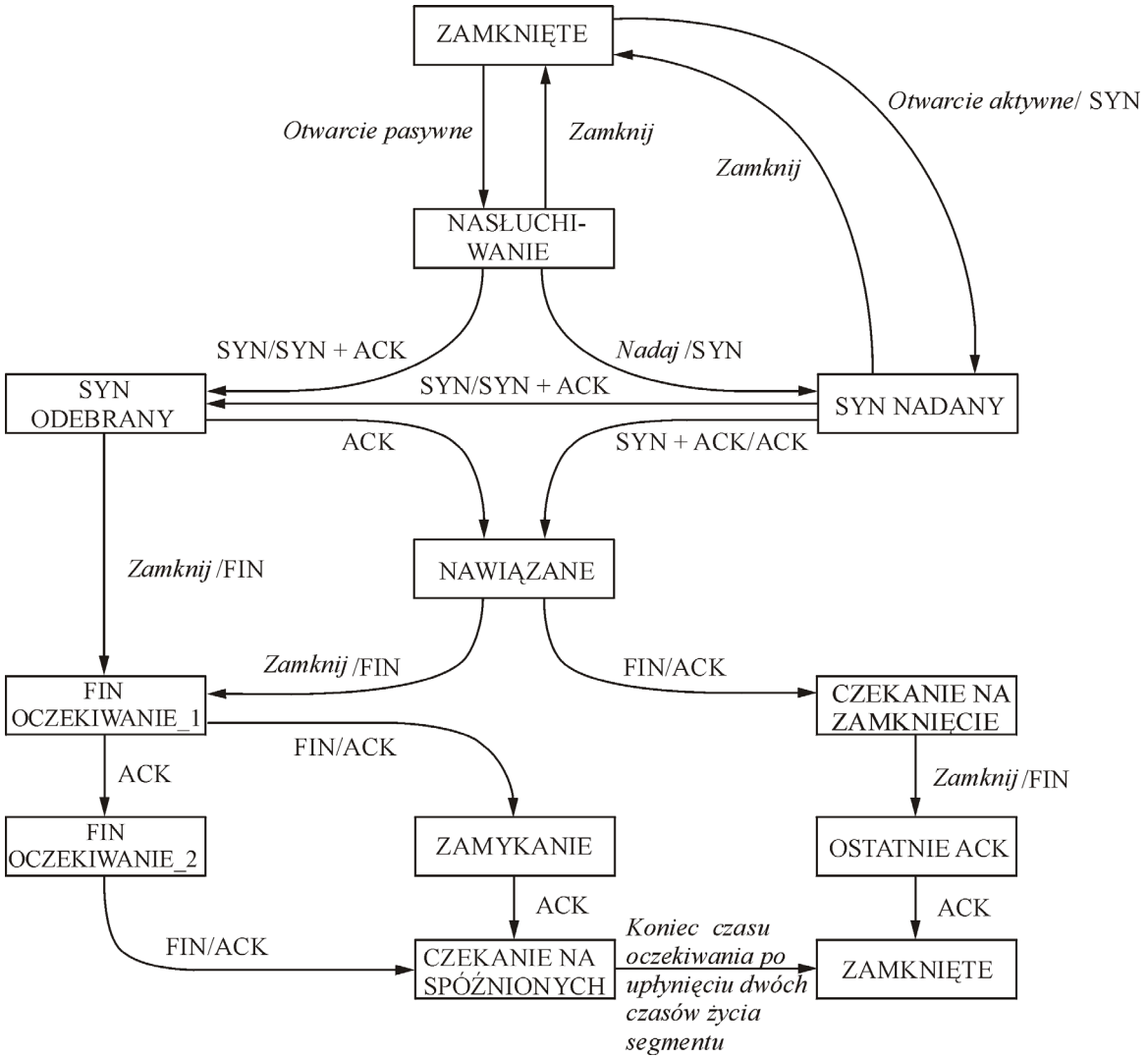
nawiązanie połączenia

---

faza danych

---

rozłączenie połączenia

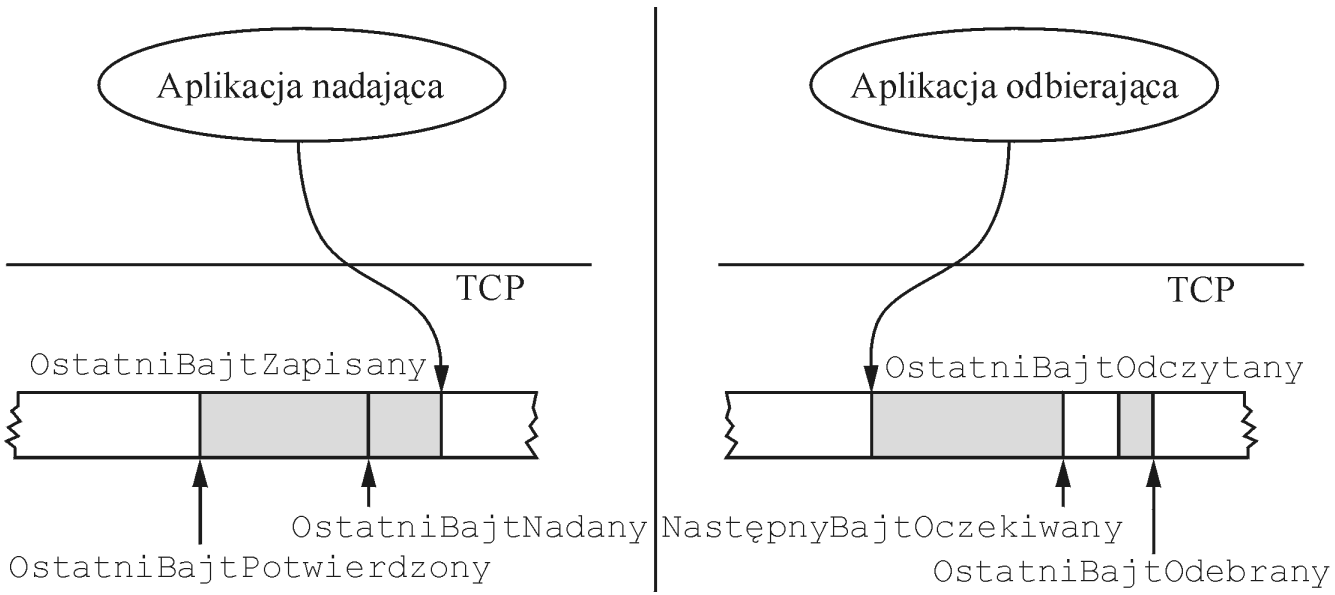


sterowanie przepływem

- *niezawodna i uporządkowana dostawa*

bufor TCP nadawczy: dane zapisane przez aplikacje, ale nienadane, dane nadane ale niepotwierdzone

bufor TCP odbiorczy: dane, których aplikacja jeszcze nie odczytała, przychodzące w kolejności i poza kolejnością



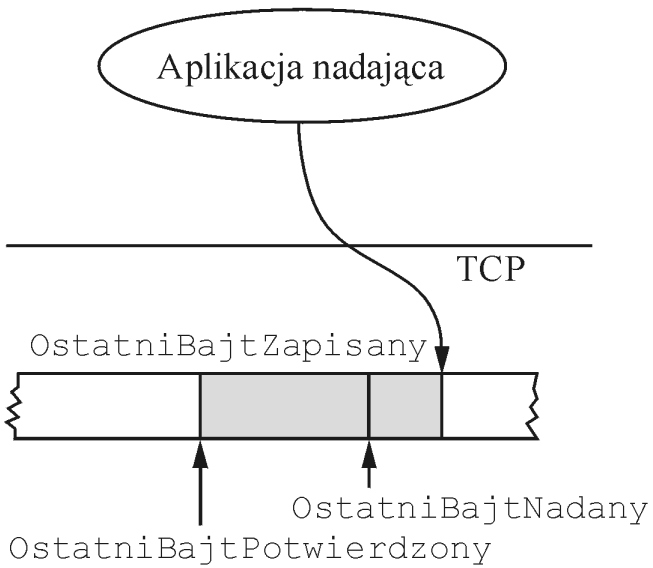
sterowanie przepływem - nadajnik

OstatniBajtPotwierdzony ≤ OstatniBajtNadany

(odbiorca nie mógł potwierdzić nie nadanego bajtu)

OstatniBajtNadany ≤ OstatniBajtZapisany

(nie można nadać bajtu nie zapisanego przez aplikację)



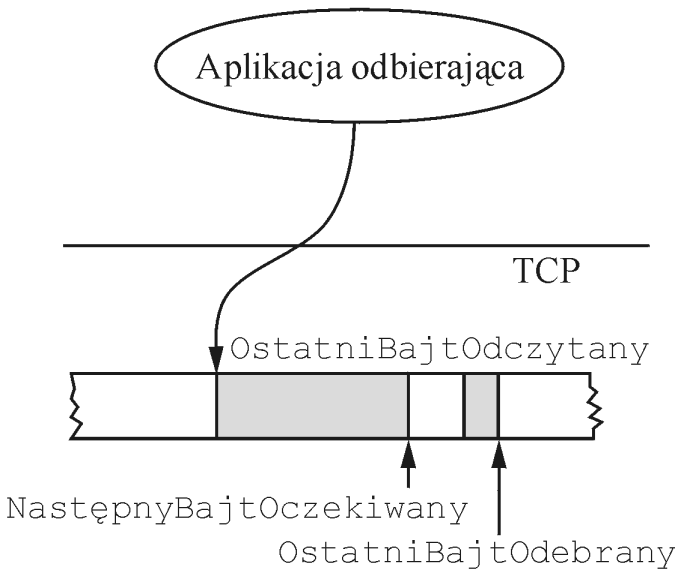
sterowanie przepływem - odbiornik

OstatniBajtOdczytany < NastępnyBajtOczekiwany

(bajt nie może być odczytany, zanim nie zostanie odebrany on i wszystkie poprzedzające go bajty)

NastępnyBajtOczekiwany ≤ OstatniBajtOdebrany + 1

- (gdy dane przyszły w kolejności, to następny bajt oczekiwany to następny po ostatnim odebranym)
- (gdy dane przyszły poza kolejnością, to następny bajt oczekiwany wskazuje na początek pierwszej przerwy w danych)



TCP po stronie odbiorczej

aby uniknąć przepełnienia bufora odbiorczego:

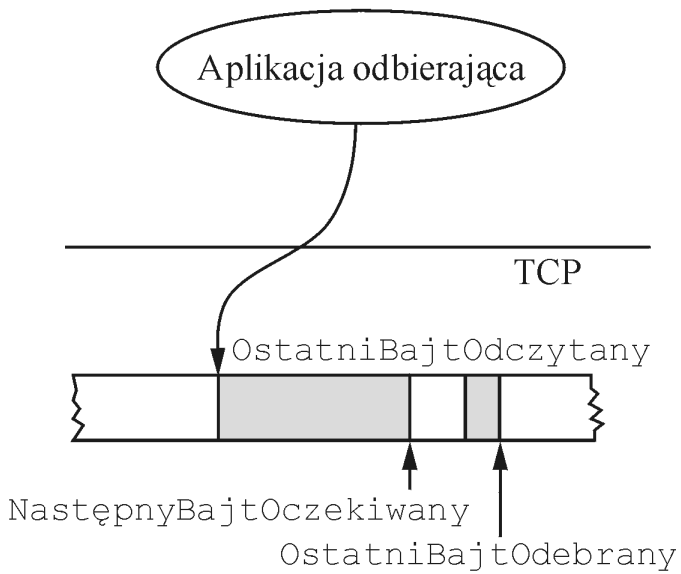
OstatniBajtOdebrany - OstatniBajtOdczytany ≤

≤ MaksymalnyBuforOdbiorczy

stąd ilość wolnego miejsca w buforze odbiorczym:

ZalecaneOkno = MaksymalnyBuforOdbiorczy -

- (OstatniBajtOdebrany - OstatniBajtOdczytany)



TCP po stronie nadawczej

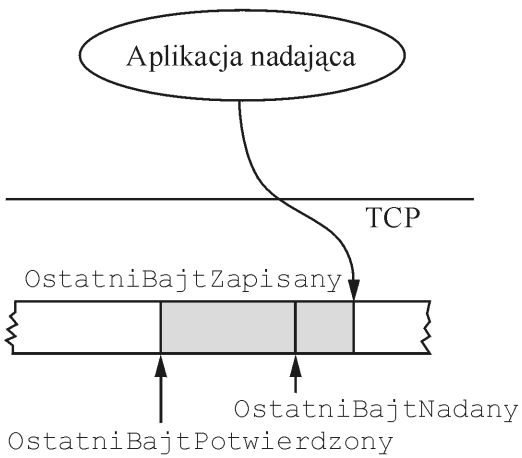
TCP po stronie nadawczej dopasowuje się do zalecanego okna: OstatniBajtNadany - OstatniBajtPotwierdzony ≤

≤ ZalecaneOkno

efektywne okno podaje, jak wiele danych nadawca może nadać: EfektywneOkno = ZalecaneOkno -

- (OstatniBajtNadany - OstatniBajtPotwierdzony)

EfektywneOkno > 0 zanim nadawca może nadawać dane



TCP po stronie nadawczej

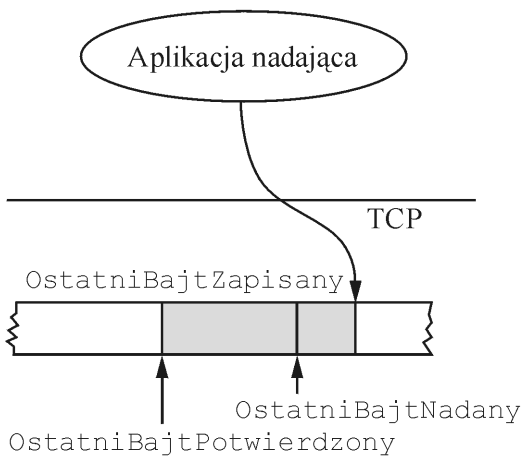
TCP po stronie nadawczej musi upewnić się, że lokalny proces aplikacji nie przepełni bufora nadawczego:

$$\text{OstatniBajtZapisany} - \text{OstatniBajtPotwierdzony} \leq \leq \text{MaksymalnyBuforNadawczy}$$

kiedy proces nadający usiłuje zapisać y bajtów do TCP, a:

$$(\text{OstatniBajtZapisany} - \text{OstatniBajtPotwierdzony}) + y > > \text{MaksymalnyBuforNadawczy}$$

wtedy TCP blokuje proces nadający



sterowanie przepływem - podsumowanie

- *wolny proces odbierający blokuje szybki proces nadający*:

1. Bufor odbiorczy się wypełnia
2. Zalecane okno kurczy się do zera
3. Strona nadawcza nie nadaje
4. Bufor nadawczy się wypełnia
5. TCP blokuje proces nadający...
6. Proces odbierający zaczyna czytać dane
7. TCP po stronie odbiorczej ponownie otwiera okno
8. Strona nadawcza nadaje dane z bufora...

wypełnianie całej pojemności łącza

- pole numer sekwencyjny w TCP ma 32 bity
- pole zalecane okno w TCP ma 16 bitów
- wymaganie algorytmu przesuwnego okna aby przestrzeń numerów sekwencyjnych była dwa razy większa od rozmiaru okna
- wymaganie spełnione: $2^{32} >> 2 \cdot 2^{16}$
- wymaganie Internetu: ten sam numer sekwencyjny nie pojawi się w okresie 60 sekund
- numer się najszybciej wyczerpie, gdy założyć, że wypełnia się całą szerokość pasma

czas upływający do wyczerpania się przestrzeni 32-bitowych numerów sekwencyjnych

szerokość pasma	Czas wyczerpania się...
T1 (1,5 Mb/s)	6,4 godziny
Ethernet (10 Mb/s)	57 minut
T3 (45 Mb/s)	13 minut
FDDI (100 Mb/s)	6 minut
STS-3 (155 Mb/s)	4 minuty
STS-12 (622 Mb/s)	55 sekund
STS-24 (1,2 Gb/s)	28 sekund

wymagana szerokość okna dla RTT wynoszącego 100 ms

- okno musi być otwarte na tyle, aby pozwolić na transmisję danych o ilości odpowiadającej iloczynowi opóźnienie $\times$ szerokość pasma
- pole zalecane okno 16 bitowe pozwala na przesłanie 2^{16} bajtów, czyli ok. 64kB

wymagana szerokość okna dla RTT wynoszącego 100 ms

szerokość pasma	Opóźnienie × szerokość pasma
T1 (1,5 Mb/s)	18 kB
Ethernet (10 Mb/s)	122 kB
T3 (45 Mb/s)	549 kB
FDDI (100 Mb/s)	1,2 MB
STS-3 (155 Mb/s)	1,8 MB
STS-12 (622 Mb/s)	7,4 MB
STS-24 (1,2 Gb/s)	14,8 MB

retransmisja adaptacyjna

- wybór czasu oczekiwania jako funkcji RTT w protokole TCP
- algorytm podstawowy:
- TCP nadając segment i odbierając ACK zapisuje ich czasy a ich różnicę określa próbkąRTT

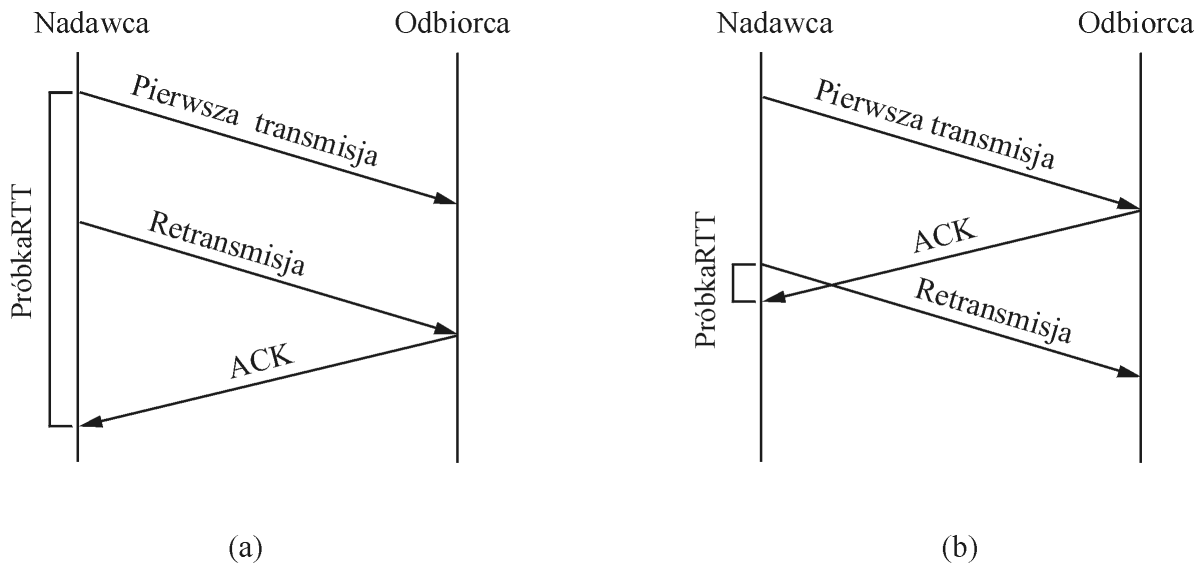
$$szacowanyRTT = \alpha \cdot szacowanyRTT + \beta \cdot próbkaRTT$$

- gdzie $\alpha + \beta = 1$, α od 0,8 do 0,9, β od 0,1 do 0,2

$$CzasOczekiwania = 2 \cdot SzacowanyRTT$$

retransmisja adaptacyjna

- algorytm Karna/Partridge'a: kiedy TCP retransmituje segment, wtedy przerywa pobieranie próbek RTT
- przyczyna: zakładano, że pomiar odnosi się do (a) pierwszej ((b) drugiej) transmisji a faktycznie odnosił się do drugiej (pierwszej), wtedy próbkaRTT jest zbyt: duża (mała)



retransmisja adaptacyjna

- algorytm Jacobsona/Karelsa

$$\text{Różnica} = \text{PróbkaRTT} - \text{SzacowanyRTT}$$

$$\text{SzacowanyRTT} = \text{SzacowanyRTT} + (\delta \cdot \text{Różnica})$$

$$\text{Odchylenie} = \text{Odchylenie} + \delta \cdot (|\text{Różnica}| - \text{Odchylenie})$$

gdzie δ między 0 i 1

$$\text{CzasOczekiwania} = \mu \cdot \text{SzacowanyRTT} + \phi \cdot \text{Odchylenie}$$

gdzie $\mu = 1$, $\phi = 4$

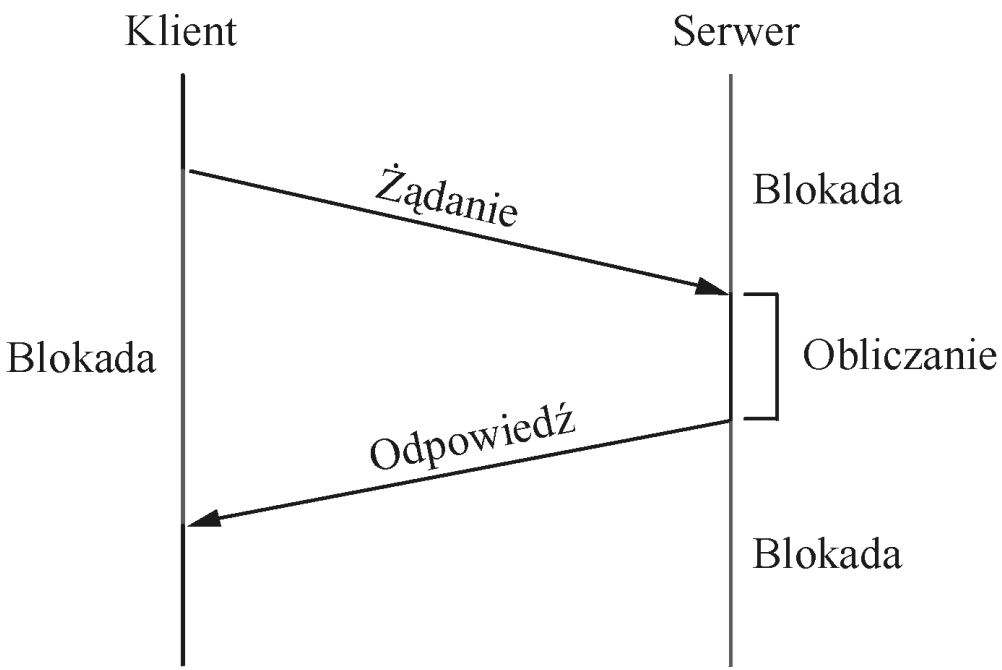
czyli gdy Odchylenie jest duże, to ono dominuje w czasie oczekiwania, gdy małe to dominuje SzacowanyRTT

granice rekordów

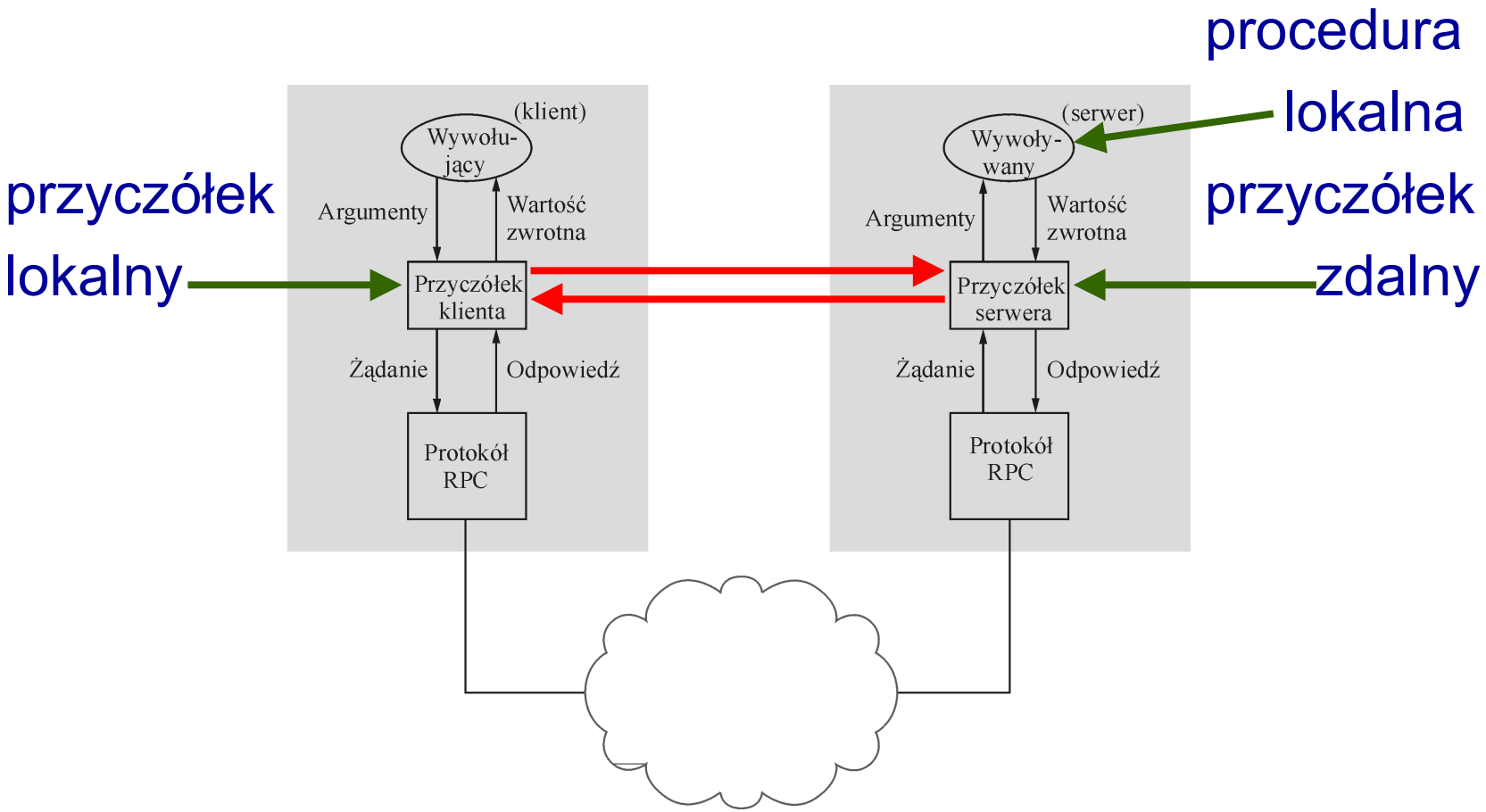
- TCP protokołem strumienia bajtów, liczba bajtów zapisanych przez nadawcę nie musi być taka sama jak liczba bajtów odczytanych przez odbiorcę
- (w UDP jest inaczej - komunikat nadany i odebrany mają taką samą długość)
- granice rekordów w TCP na dwa sposoby:
- flaga PUSH w nagłówku TCP
- flaga URG w nagłówku TCP

zdalne wywołanie procedury

- paradygmat *żądanie/odpowiedź* (transakcja komunikatów)
- *model klienta i serwera* (warstwa sesji)

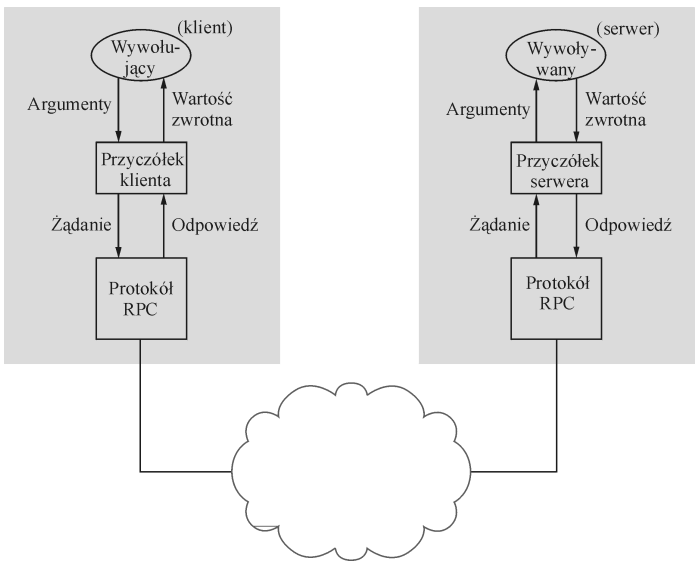


zdalne wywołanie procedury



zdalne wywołanie procedury

- protokół zarządzający komunikatami między procesami klienta i serwera
- język programowania i kompilator, pozwalający na umieszczenie argumentów w komunikacie żądania w komputerze klienta i odtworzenie tych argumentów w komputerze serwera

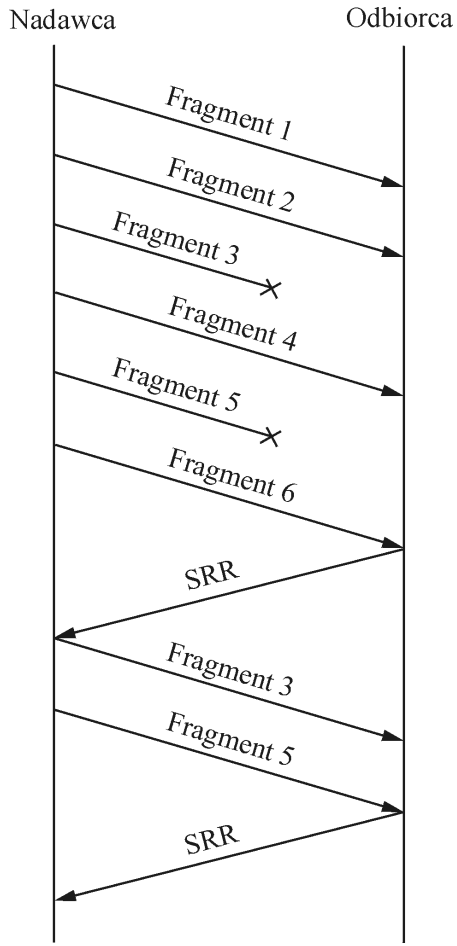


protokół RPC

- *mikroprotokoły:*
- BLAST: dzieli na fragmenty i składa komunikaty
- CHAN: synchronizuje komunikaty żądania i odpowiedzi
- SELECT: rozdziela komunikaty żądań do właściwych procesów

przesyłanie masowe (BLAST)

- *nadawca* dzieli duży komunikat na fragmenty i przesyła je jeden za drugim
- *odbiorca* nadaje selektywne żądanie retransmisji (SRR) tych fragmentów, które nie dotarły
- gdy przyszły wszystkie, SRR potwierdza cały komunikat
- BLAST zastępuje tu protokół transportowy?



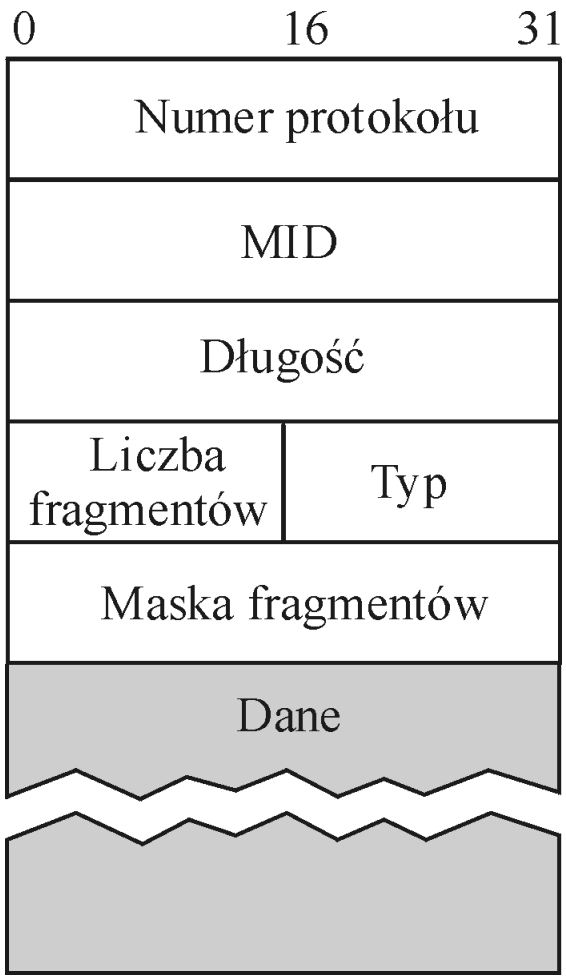
format komunikatu w protokole BLAST

- numer protokołu (nad BLAST)
- identyfikator komunikatu MID

(fragment należy do komunikatu *m*)

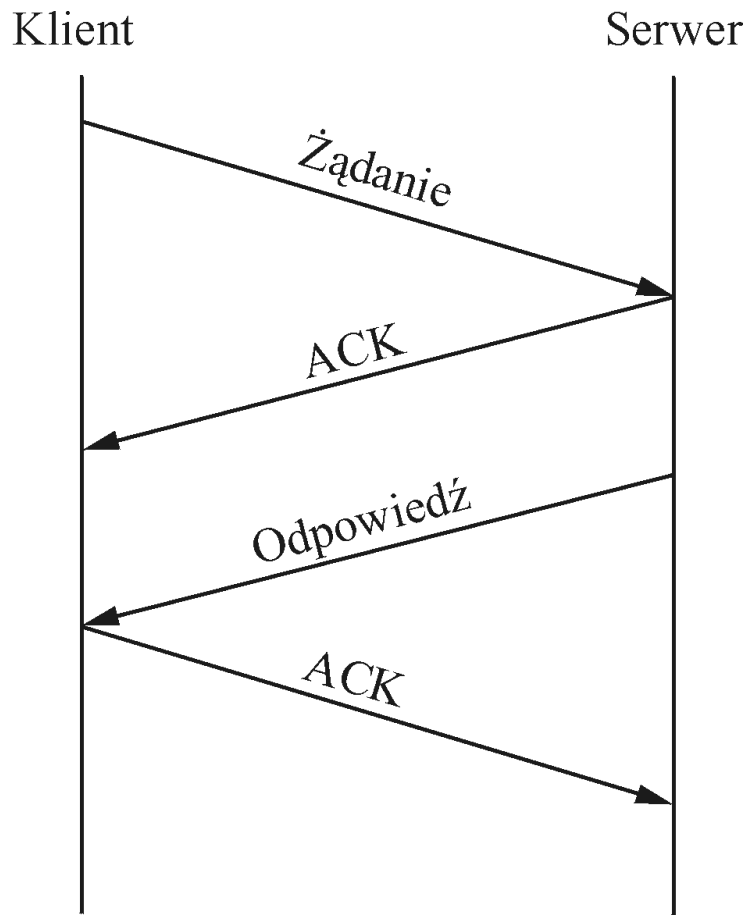
- długość danych fragmentu
- typ: *dane* albo SRR
- maska fragmentów (32 bity):

dla danych, *i*-ty bit=1 to *i*-ty fragment,
dla SRR, *i*-ty bit=1 oznacza, że
właśnie przyszedł fragment *i*,
i-ty bit=0 oznacza, że fragment *i*
został stracony



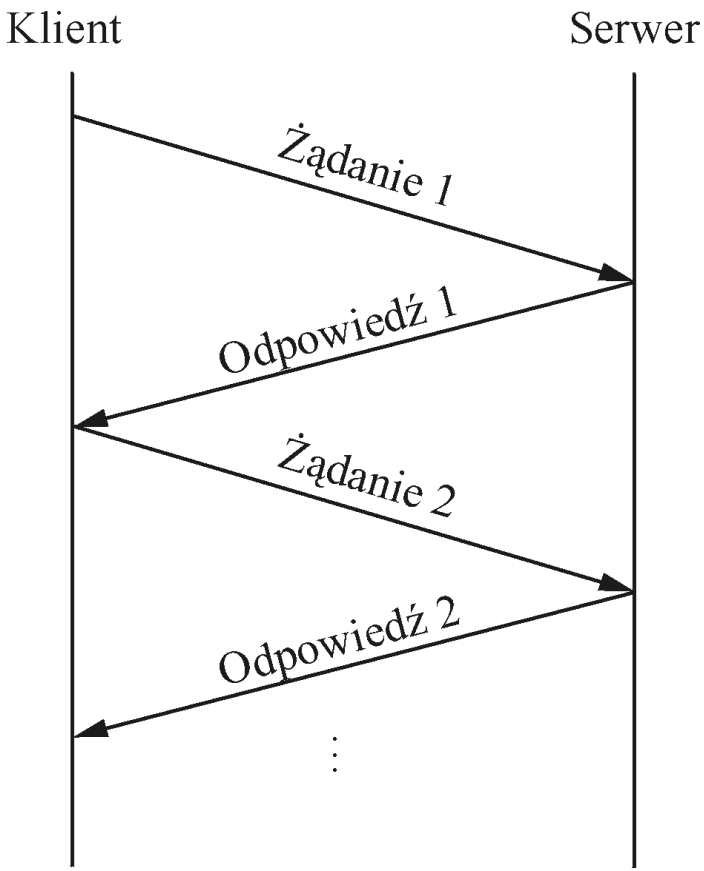
żądanie/odpowiedź (CHAN)

- implementuje kanał logiczny
- semantyka *najwyżej raz* (najwyżej jedno żądanie)
- algorytm:
- klient nadaje żądanie, serwer je potwierdza
- po wykonaniu procedury: serwer nadaje odpowiedź, klient ją potwierdza



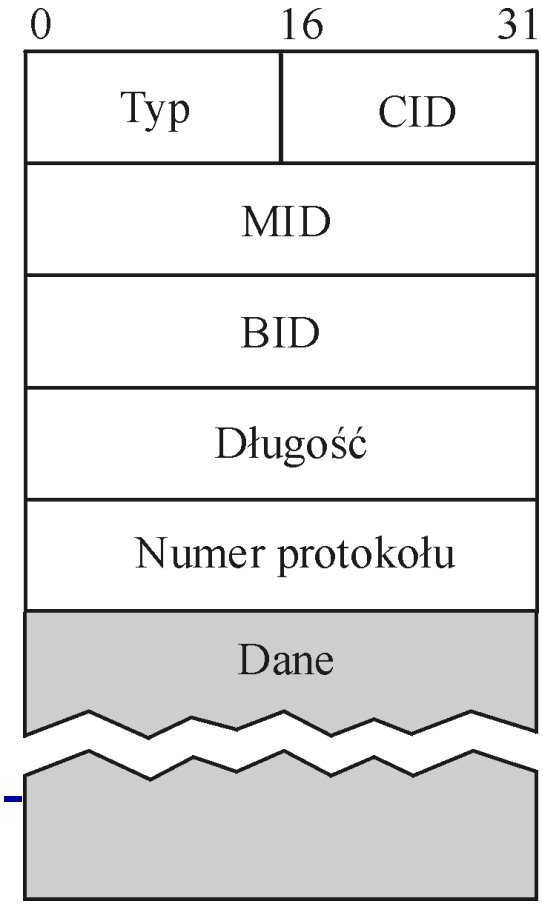
żądanie/odpowiedź (CHAN)

- ukryte potwierdzenia:
- odpowiedź1 potwierdza żądanie1,
- żądanie2 potwierdza odpowiedź1



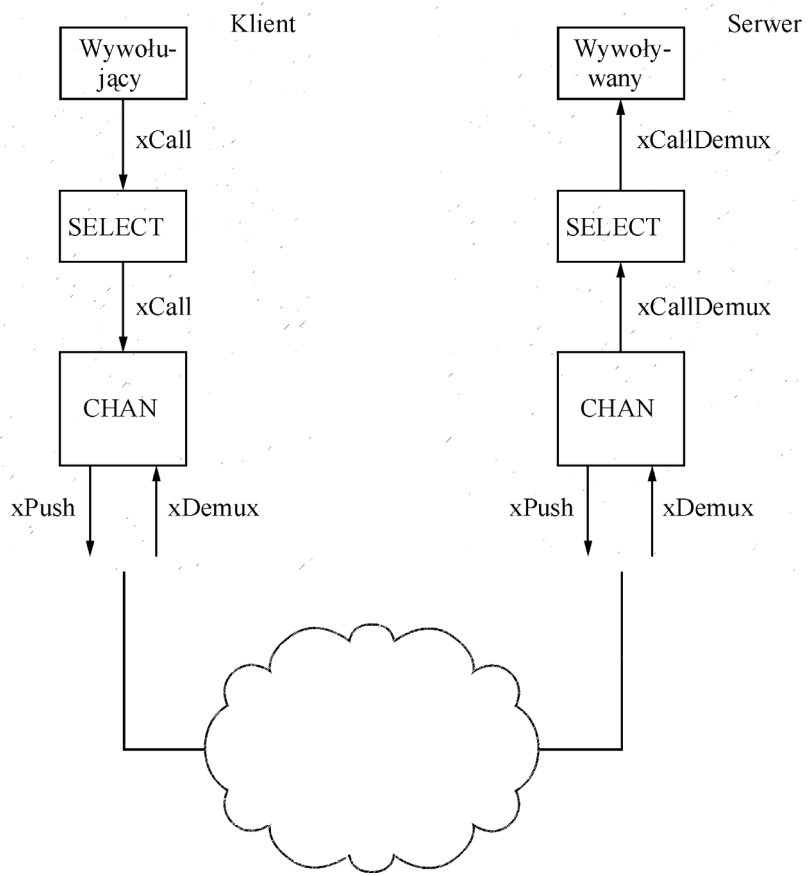
format komunikatu w protokole CHAN

- typ: REQ, REP, ACK, PROBE
  PROBE= żyjesz? (serwerze)
- CID identyfikator kanału, 2^{16}=64k współbieżnych transakcji
- MID identyfikator pary żądanie/odpowiedź
- BID identyfikator ładowania (ochrona przed starymi komunikatami u odbiorcy, spowodowanymi przeładowaniem komputera)



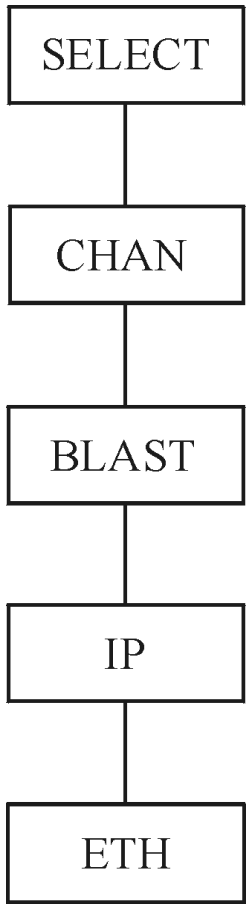
dyspozytor (SELECT)

- *rozdziela* komunikaty żądań do właściwej procedury
- odpowiada demultipleksacji w UDP
- po stronie klienta:
  SELECT otrzymuje
  numer procedury i
  przekazuje go do CHAN,
  zwraca wynik do klienta
- po stronie serwera:
  SELECT wykorzystuje
  numer procedury do
  wyboru lokalnej procedury,
  którą ma wywołać



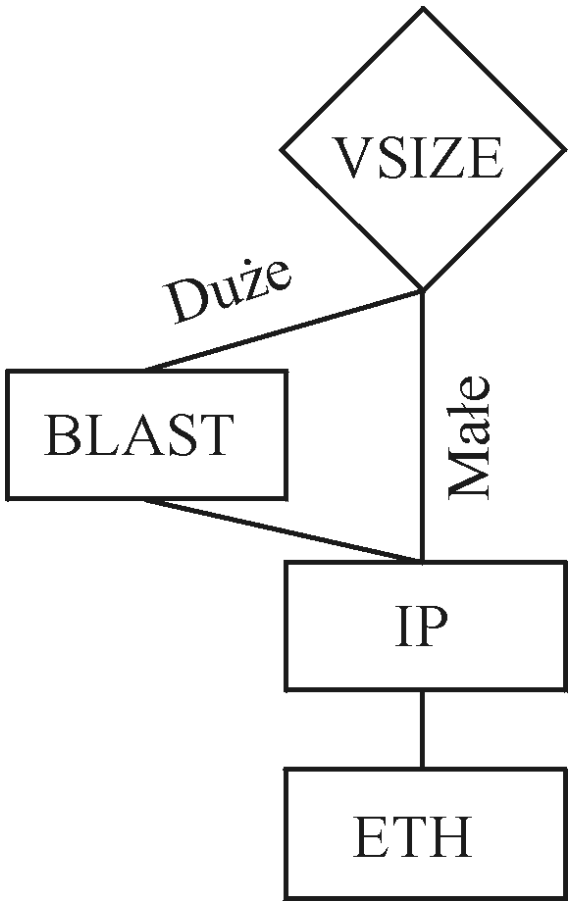
złożenie wszystkiego razem

- fragmentacja/składanie przesunięte z IP do protokołu BLAST (komunikaty do 32kB), zawodna dostawa komunikatów
- CHAN: algorytm żądanie/odpowiedź, niezawodna dostawa komunikatów
- SELECT: identyfikacja zdalnych procedur



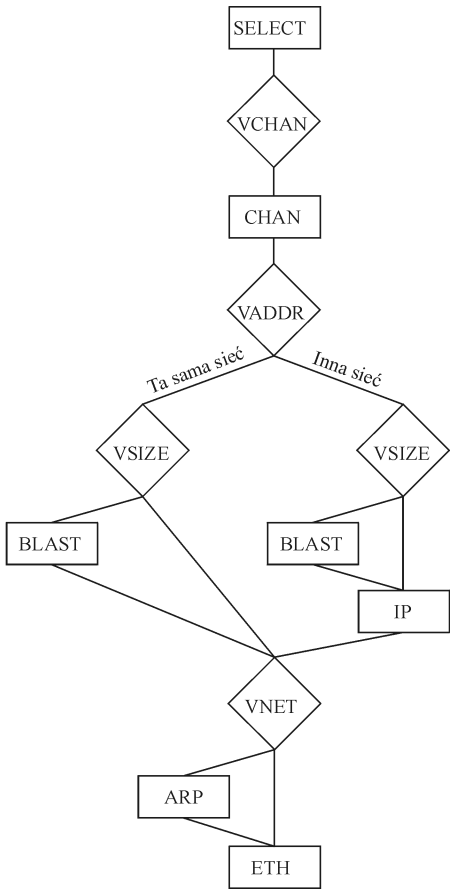
graf protokołów RPC stosujący VSIZE

- *protokół wirtualny* służy jedynie do kierowania komunikatów przez graf protokołów (n.p. protokół VSIZE inaczej kieruje komunikaty duże i małe)
- protokół wirtualny *nie porozumiewa się ze swoim odpowiednikiem* po drugiej stronie (nie dodaje więc nagłówków)



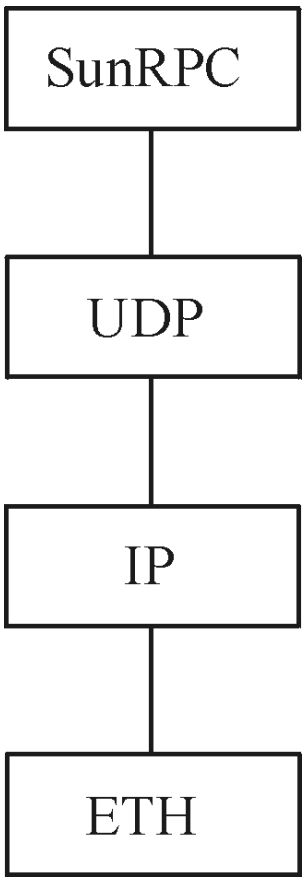
graf protokołów RPC stosujący protokoły wirtualne

- VCHAN decyduje, którą sesję CHAN wybrać (zarządzanie współbieżnością)
- VADDR decyduje czy stosować IP czy nie
- VSIZE, decyduje, czy stosować BLAST, czy nie
- VNET decyduje, czy stosować ARP czy nie



SunRPC

- rozpowszechniany z sieciowym systemem plików NFS firmy Sun
- de facto norma
- fragmentacja komunikatów
- synchronizacja komunikatów żądania i odpowiedzi
- rozdzielanie komunikatów do właściwej procedury
- SELECT podzielone między UDP i SunRPC



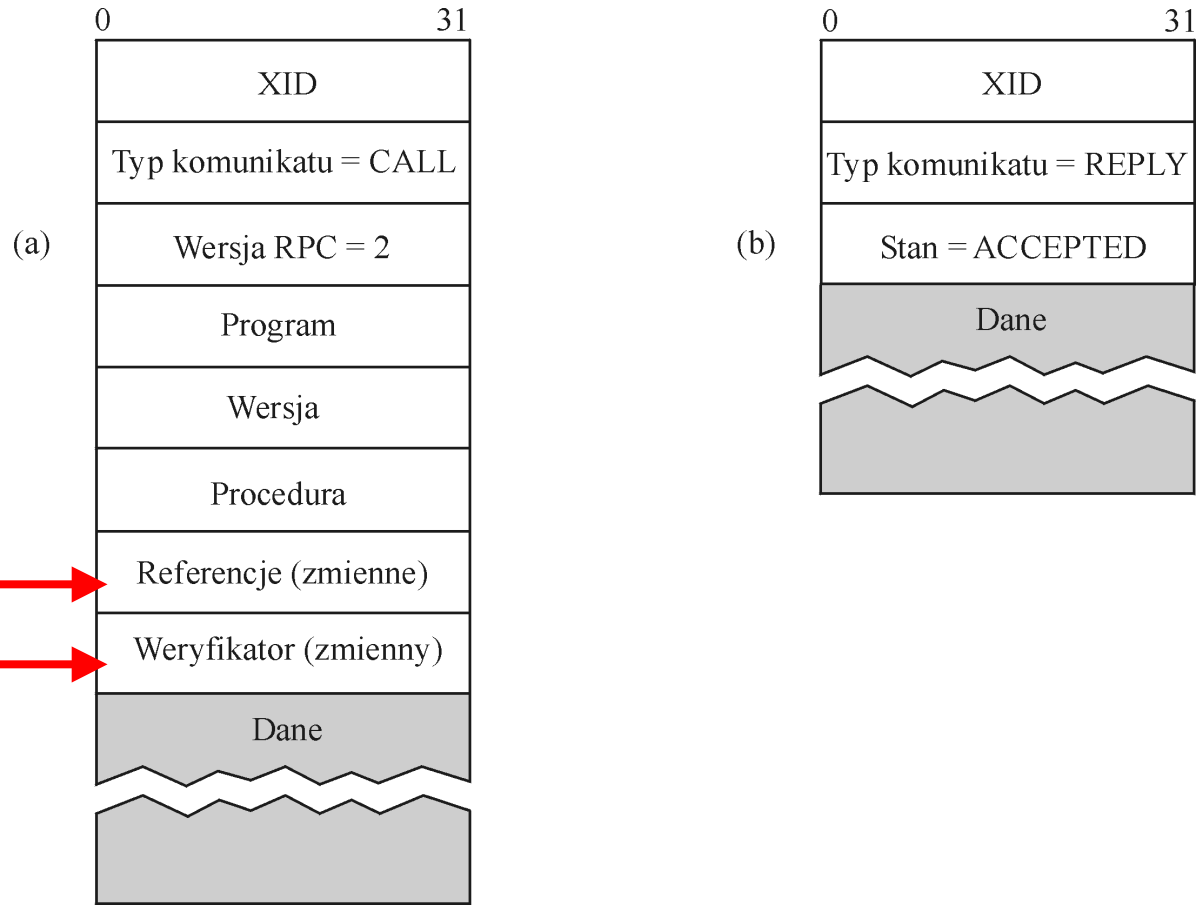
format nagłówka SunRPC

żądanie

odpowiedź

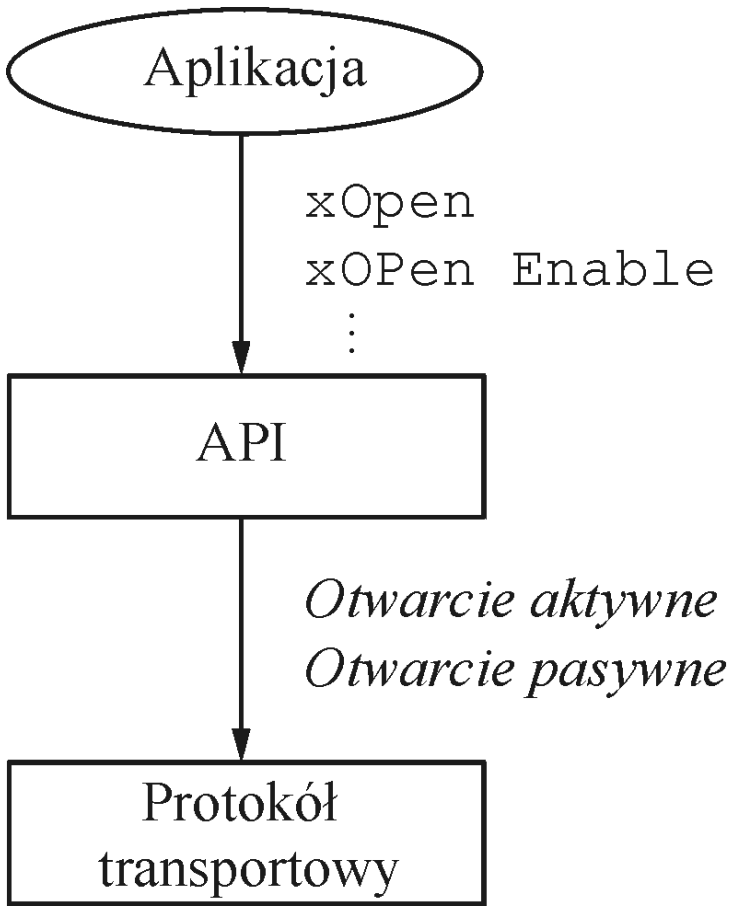
XID:
identyfikator
transakcji

uwiarygodnienie
przed serwerem



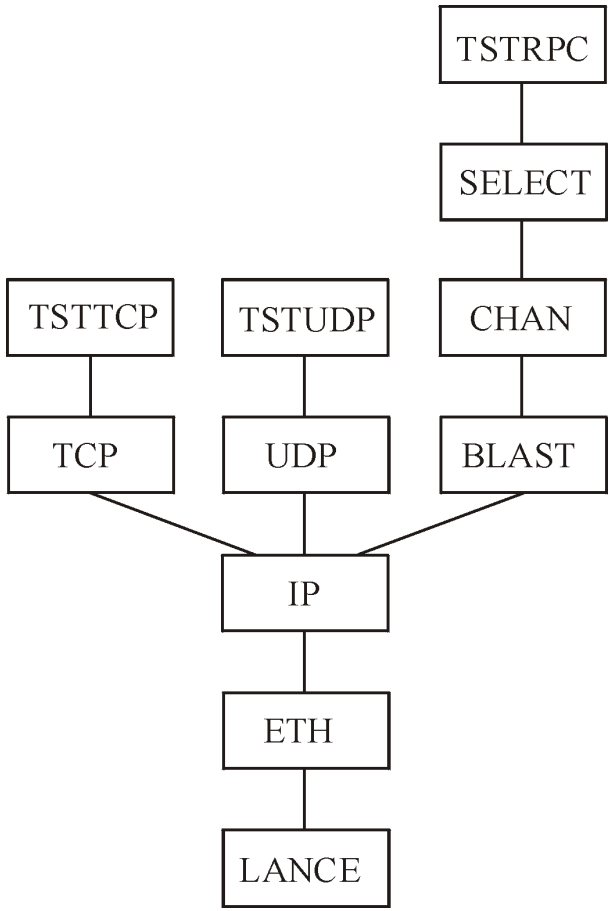
interfejs programowania aplikacji

- API - interfejs jaki protokół zapewnia programowi aplikacji
- interfejs gniazda w Uniksie Berkeley
- operacje:
  utworzenia gniazda
  otwarcie pasywne w serwerze
  otwarcie aktywne u klienta
  (trzyetapowa wymania)
  nadawanie i odbiór danych



efektywność grafu protokołów

- stos protokołów stosowany w pomiarach efektywności



opóźnienie (RTT)

rozmiar (B) komunikatu	UDP	TCP	RPC
1	279	365	428
100	413	519	593
200	572	691	753
300	732	853	913
400	898	1016	1079
500	1067	1185	1247
600	1226	1354	1406
700	1386	1676	1566

przepustowość (UDP)

- polepsza się ze wzrostem komunikatu (płaska przy 16kB)
- maksimum ok. 9,5 Mb/s (ograniczenie adaptera sieci)