Komunikacja

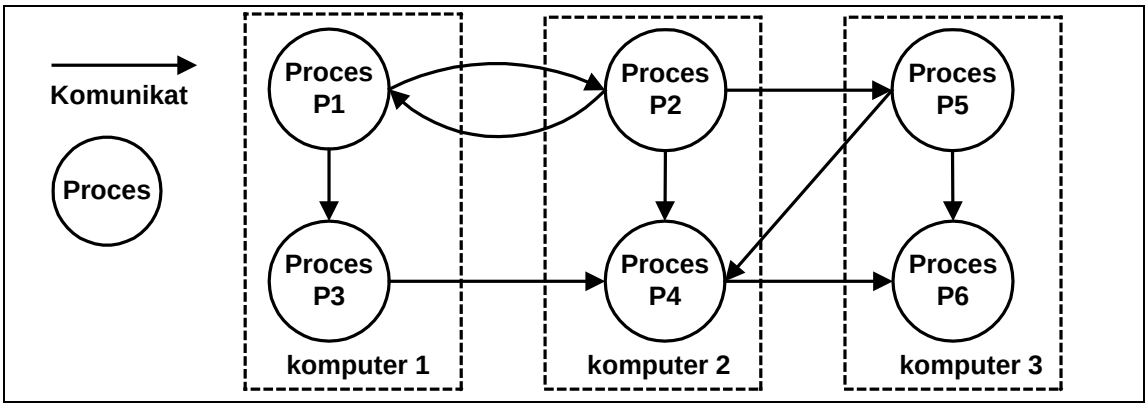
1 Podstawowe modele przetwarzania

Obecnie stosuje się następujące modele przetwarzania:

- Model procesów i komunikatów
- Model procesów komunikujących się poprzez pamięć dzieloną

Model procesów i komunikatów:
Model procesów i komunikatów skonstruowany jest w oparciu o następujące reguły:

- Aplikacja składa się ze zbioru procesów sekwencyjnych. Procesy mogą być wykonywane równolegle.
- Proces wykonuje się sekwencyjnie i używa swej pamięci lokalnej.
- Proces komunikuje się z otoczeniem za pomocą komunikatów. Są dwie podstawowe operacje komunikacyjne: wysłanie komunikatu i odbiór komunikatu.
- Procesy mogą być przydzielone do procesorów w różny sposób. Poprawność działania aplikacji nie powinna zależeć od tego podziału.



Rys. 1-1 Aplikacja współbieżna jako zbiór komunikujących się procesów

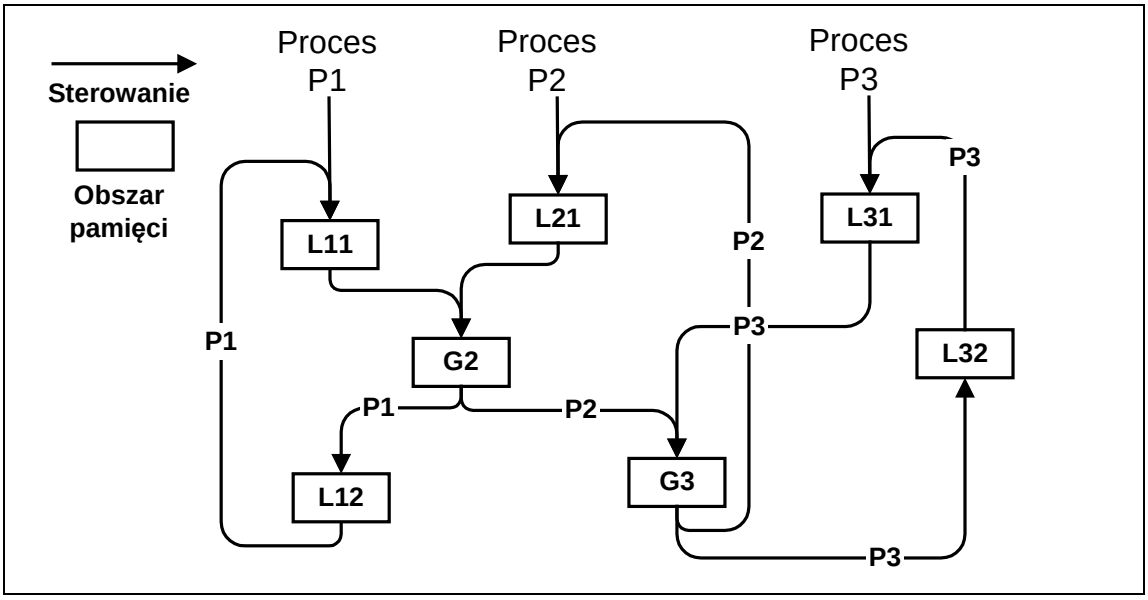
Własności modelu procesów i komunikatów:

- Wydajność - model procesu sekwencyjnego posiada wydajne narzędzia implementacyjne (kompilatory itd.). Gdy procesy wykonywane są na różnych procesorach, sprzęt jest dobrze wykorzystany.
- Niezależność od fizycznej struktury maszyny - model procesów i komunikatów jest niezależny od tego czy procesy wykonywane są w systemie jedno, wieloprocesorowym czy rozproszonym.
- Skalowalność - model jest niezależny od liczby dostępnych procesorów.
- Modularność - problem może być podzielony na tworzone niezależnie moduły (procesy), komunikujące się poprzez dobrze zdefiniowane interfejsy (komunikaty, kanały).
- Determinizm - program jest deterministyczny gdy takie same sekwencje na wejściu, powodują takie same sekwencje na wyjściu. W modelu procesów i komunikatów łatwo osiągnąć determinizm.

Model procesów komunikujących się poprzez pamięć wspólną

Model procesów komunikujących się poprzez pamięć wspólną skonstruowany jest w oparciu o następujące reguły:

- Aplikacja składa się ze zbioru procesów lub (i) wątków wykonywanych współbieżnie. Gdy sprzęt na to pozwala procesy i wątki mogą być wykonywane równolegle.
- Proces wykonuje się sekwencyjnie i używa swej pamięci lokalnej lecz może tworzyć działające współbieżnie wątki.
- Proces komunikuje się z innymi procesami za pomocą obszarów pamięci dzielonej. Gdy sprzęt na to pozwala komunikacja może zachodzić pomiędzy komputerami.
- Dostęp do pamięci dzielonej jest synchronizowany za pomocą różnych narzędzi (muteksy, semafory, itd.)
- Procesy mogą być przydzielone do procesorów w różny sposób. Poprawność działania aplikacji nie powinna zależeć od tego podziału.



Rys. 1-2 Ilustracja modelu przetwarzania z uzyciem pamięci dzielonej

Własności modelu:

- Komunikacja ogranicza się typowo do jednej maszyny
- Konieczność ochrony sekcji krytycznej przy dostępie do obszarów pamięci dzielonej
- Duża szybkość działania

2 Rodzaje komunikacji

Składowe systemu mogą być rozdzielone tak logicznie jak fizycznie. Aby mogły one współdziałać muszą się komunikować.

Aby ujednolicić podejście do komunikacji Międzynarodowa Organizacja Normalizacyjna ISO opracowała model w którym wyodrębniono poziomy na których realizowane są dobrze zdefiniowane funkcje. Jest to „Model wzorcowy połączeń w systemach otwartych” (ang. *Open Systems Interconnection Reference Model*) nazywany modelem ISO OSI.

Model opracowano oby zapewnić systemom otwartym wzajemną komunikację.

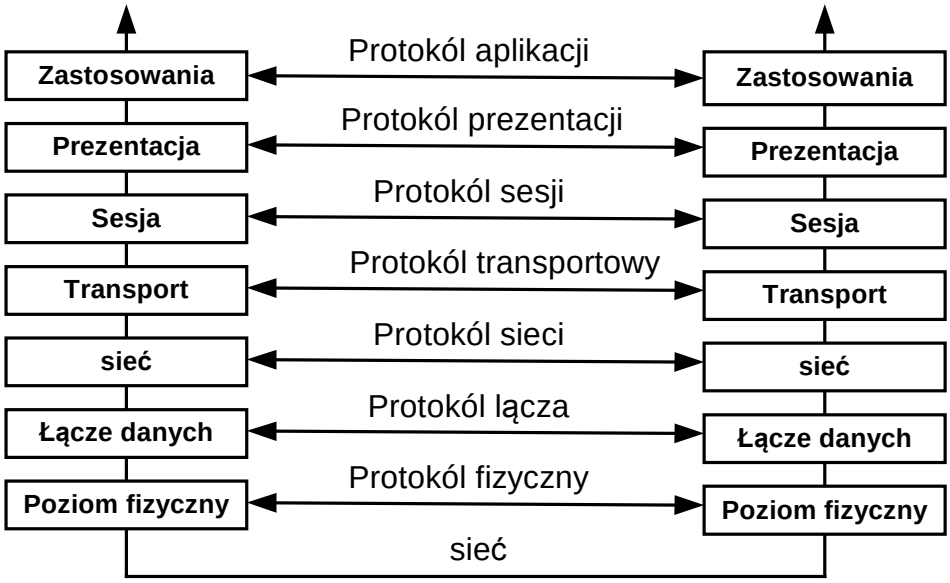
System otwarty – system z którym można się komunikować o ile komunikacja przebiega według ustalonych reguł definiujących format, treść i znaczenie odbieranych i wysyłanych komunikatów

Protokół - reguła specyfikująca format, treść i znaczenie komunikatów

W modelu OSI komunikację podzielono na warstwy z których każda pełni określoną funkcję. Każda warstwa dostarcza interfejsu warstwie znajdującej się powyżej.

Interfejs specyfikuje:

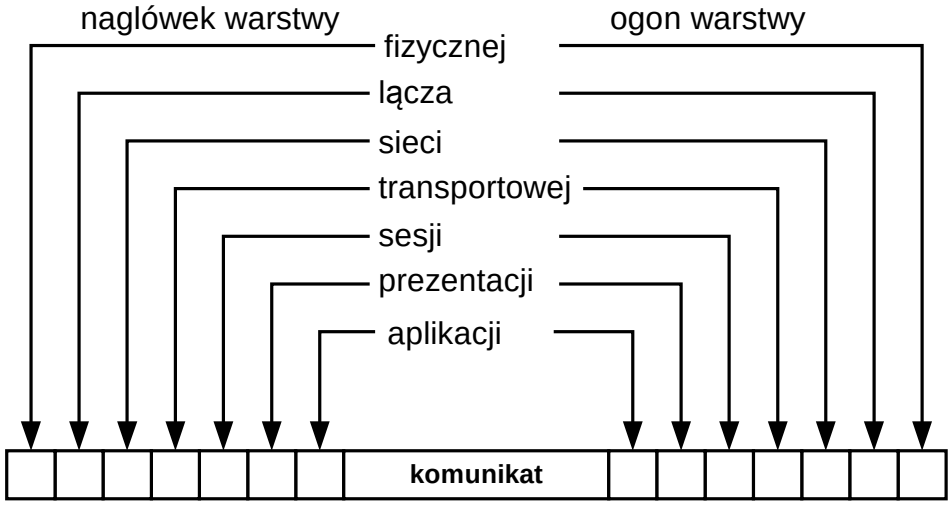
- Zestaw usług świadczonych przez daną warstwę
- Sposób dostępu do tych usług



Stos protokołów komunikacji w modelu ISO OSI

Gdy proces działający na maszynie M1 chce skorzystać z usługi procesu działającego na maszynie M2 w tej samej warstwie to:

- buduje komunikat i przekazuje go do warstwy niższej
- komunikat przechodzi do kolejnych niższych warstw i jest obudowywany nagłówkami i ogonami
- trafia do maszyny M2
- przechodzi do kolejnych warstw wyższych i jest pozbawiany nagłówków i ogonów

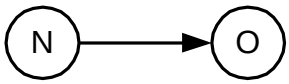


- Warstwa fizyczna – przekazuje zera i jedynki, definiuje standard mechaniczny, poziomy napięć kierunki transmisji.
- Warstwa łącza – przekazuje bity. Zadanie: zapewnienie wolnego od błędów przesłania bitów. Osiągane poprzez grupowanie bitów w ramki uzupełnione sumami kontrolnymi. Przykład: Ethernet
- Warstwa sieciowa – przekazuje ramki od źródła do przeznaczenia, zadania - znajdowanie i wytyczanie trasy. Przykład: protokół IP
- Warstwa transportowa – przekazuje bajty – zadanie bezbłędnie przekazać bajty – grupowanie, przesyłanie, zapewnianie właściwej kolejności, retransmisja pakietów błędnych. Przykład: TCP
- Warstwa sesji – funkcja kontrola dialogu, rzadko używana.
- Warstwa prezentacji jest odpowiedzialna za zarządzanie sposobem kodowania wszelkich danych.
- Pełni rolę interfejsu pomiędzy aplikacjami użytkownika a usługami sieci. Przykład: protokół FTP, HTTP

Klasyfikacje komunikacji:

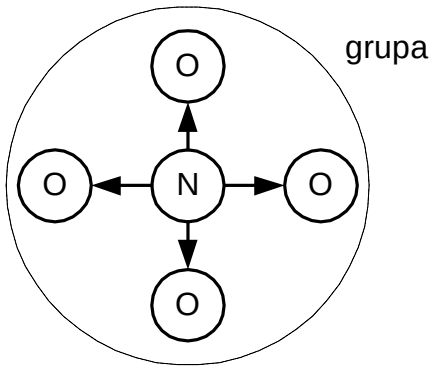
Rodzaje komunikacji ze względu na liczbę odbiorców

- Punkt - punkt
- Komunikacja grupowa



jeden do jednego

N - nadawanie
O - odbiór



jeden do wielu

Grupa – zbiór procesów działających wspólnie w sposób określony poprzez system lub użytkownika.

- Komunikacja punkt – punkt jest powszechnie stosowana w modelu klient serwer
- Komunikacja grupowa stosowana w systemach rozproszonych (tolerowanie uszkodzeń, przetwarzanie równoległe, wyszukiwanie informacji, zwielokrotnione aktualizacje)

Rodzaje komunikacji ze względu na trwałość uczestników komunikacji:

- Komunikacja trwała
- Komunikacja nietrwała

Rodzaje komunikacji ze względu na ciągłość danych:

- Dwukierunkowy strumień (*ang. stream*) – brak separacji danych, przykład: TCP
- Komunikat (ang. *message*) – dane są separowane, przykład: UDP

W praktyce najczęściej stosowane są następujące rodzaje komunikacji:

- Komunikacja oparta na komunikatach
- Zdalne wywoływanie procedur – Sun RPC
- Zdalne wywoływanie metod – JavaRMI, CORBA
- Komunikacja strumieniowa

3 Komunikaty

Przesłanie komunikatu pomiędzy procesami jest przesłaniem pomiędzy nimi pewnej liczby bajtów według ustalonego protokołu. Przesłanie komunikatu jest operacją atomową.

Komunikacja między parą procesów obejmuje działania po stronie procesu nadawczego i odbiorczego. Są to:

- Przenoszenie danych
- Synchronizacja

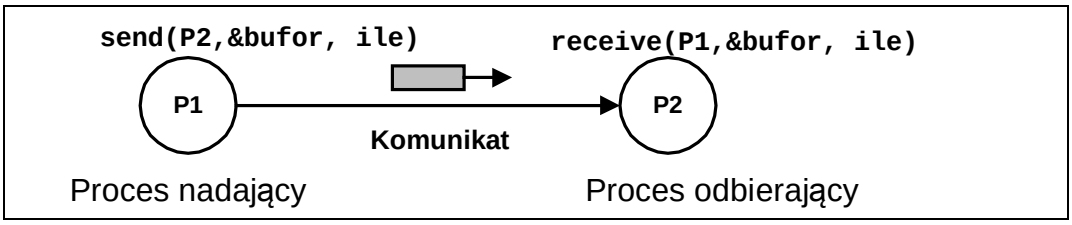
Z przesyłaniem komunikatów wiążą się następujące problemy:

- problem synchronizacji nadawcy i odbiorcy (kto i kiedy czeka),
- problem trwałości nadawcy i odbiorcy
- problem adresowania (jaki system adresacji),
- problem identyfikacji (czy procesy znają swoje identyfikatory),
- problem przepływu danych (w jedną czy dwie strony),
- problem zapewnienia niezawodnego przesłania przez zawodny kanał komunikacyjny.
- Problem implementacji

Do zapewnienia komunikacji potrzebne są przynajmniej dwie funkcje interfejsowe – wysyłająca i odbierająca komunikat.

Wysłanie komunikatu: **`send(id_odb, void *bufor, int ile)`**

Odbiór komunikatu: **`receive(id_nad, void *bufor, int ile)`**

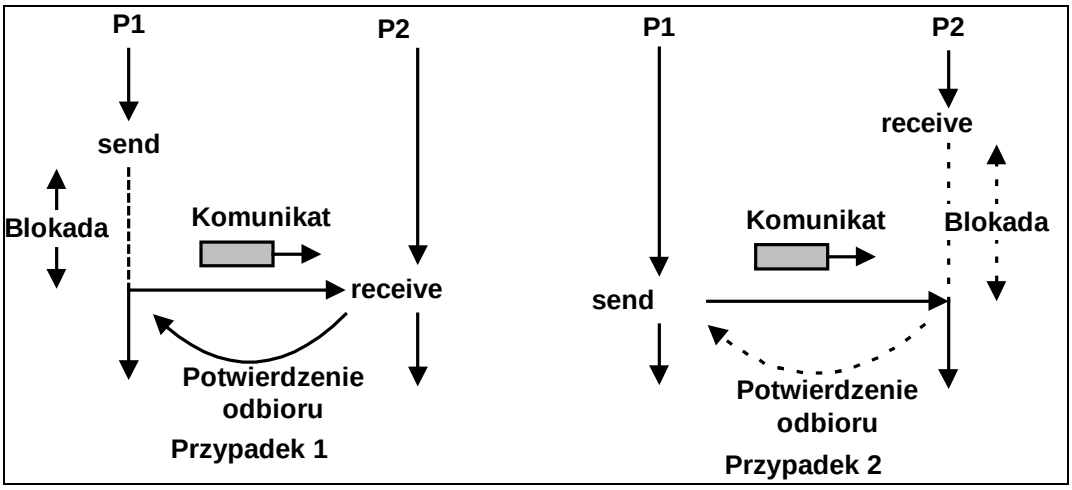


Przesyłanie komunikatów pomiędzy procesami P1 i P2

Problem synchronizacji: Komunikacja synchroniczna i asynchroniczna

Komunikacja synchroniczna

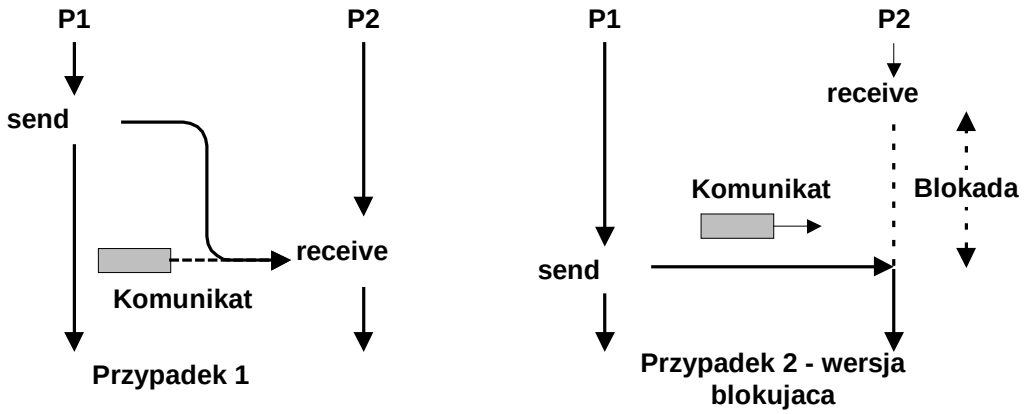
- Proces wysyłający jest blokowany do czasu otrzymania potwierdzenia że proces docelowy otrzymał wysyłany komunikat
- Gdy w momencie wykonania funkcji **receive** brak jest oczekującego komunikatu, proces odbierający jest wstrzymywany do czasu nadejścia jakiegoś komunikatu. Gdy jakiś komunikat oczekuje, proces odbierający nie jest blokowany.



Rys. 3-1 Komunikacja synchroniczna pomiędzy procesami P1 i P2

Komunikacja asynchroniczna

- Proces wysyłający komunikat nie jest blokowany.
- Proces odbierający jest wstrzymywany do czasu nadejścia komunikatu (wersja blokująca) lub też nie jest wstrzymywany (wersja nie blokująca). Informacja czy odebrano komunikat czy też nie, przekazywana jest jako kod powrotu funkcji odbierającej.



Rys. 3-2 Komunikacja asynchroniczna pomiędzy procesami P1 i P2

Rodzaj komunikacji	Blokada przy wysyłaniu	Blokada przy odbiorze
Synchroniczna	Tak	Tak
Asynchroniczna	Nie	Tak - wersja blokująca Nie - wersja nie blokująca

Tab. 1 Definicja komunikacja synchronicznej i asynchronicznej

Buforowanie

Przy transmisji asynchronicznej konieczne jest buforowanie po stronie wysyłającej. Powstaje pytanie:

- Jaka powinna być pojemność tego bufora ?
- Co zrobić gdy bufor się przepełni ?

Postępowanie w przypadku przepełnienia bufora:

- Zablokować proces wysyłający.
- Funkcja wysyłająca komunikat kończy się błędem.

	Synchroniczna	Asynchroniczna
Obsługa błędów	Testowanie kodu powrotu	Konieczna obsługa wyjątków
Buforowanie po stronie wysyłającej	Nie	Tak
Szybkość przetwarzania	Mniejsza	Większa

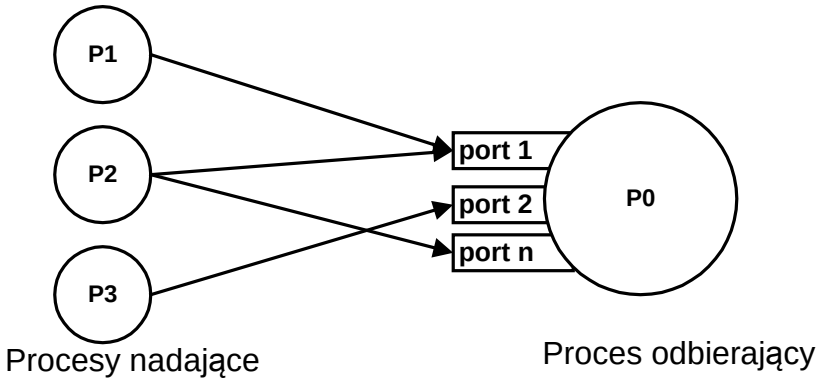
Porównanie komunikacji synchronicznej i asynchronicznej

Adresowanie

Przeźroczystość położenia

W systemach rozproszonych dąży się do uzyskania przeźroczystości położenia adresów. Adresowanie odbywa się w sposób symboliczny. Odwzorowanie adresu symbolicznego na fizyczny wykonywane jest przez odpowiednie warstwy oprogramowania. Usługi mogą być przemieszczane bez powiadamiania klientów.

Port:
Jeden z wielu możliwych wejść do procesu odbiorczego, miejsce przeznaczenia komunikatu. Z portem związana może być związana kolejka lub nie być.

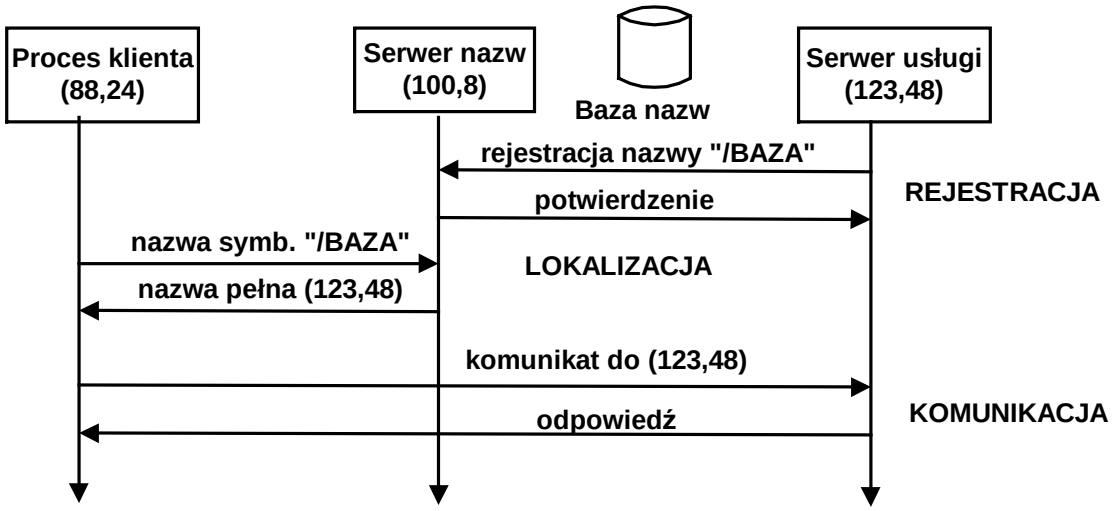


Do adresowania procesu docelowego stosuje się następujące rozwiązania:

1. PID procesu
2. Porty
3. Gniazda (IP maszyny + numer portu)
4. Grupy procesów
5. Grupy portów
6. Nazwy symboliczne
7. Obiekty

Adres	System operacyjny	Przeźroczystość położenia
PID procesu	V	Tak
Porty	Mach, Chorus, Amoeba	Tak
Gniazda	BSD 4, Unix	Nie
Grupy procesów	V, Amoeba	Tak
Grupy portów	Chorus	Tak
Nazwy symboliczne	QNX	Tak
Obiekty	Clouds, Emerald, Java RMI	Tak

Aby zastosować nazwy symboliczne konieczny jest usługa nazewnicza (ang. *Name service*).



Komunikacja procesu klienta z serwerem usługi „/BAZA”

Powstaje pytanie skąd znana jest lokalizacja serwera nazw ?.

1. Lokalizacja serwerów nazw jest ustalona i podana jako parametr instalacyjny systemu
2. Serwery nazw rozgłaszają swą lokalizację innym węzłom.

Identyfikowanie procesów

We wzajemnej komunikacji procesów występuje problem identyfikacji procesów.

Przykłady:

1. Połączenie telefoniczne – dzwoniący musi znać numer tego do kogo dzwoni ale odbiór nie wymaga znajomości dzwoniącego.
2. Tablica ogłoszeń – każdy może umieścić ogłoszenie i każdy może je odczytać lub zdjąć

W praktyce stosowane są różne rozwiązania dotyczące identyfikacji komunikujących się procesów:

1. Kanał symetryczny (dedykowany) – aby się skomunikować procesy muszą znać swe identyfikatory (Occam).
2. Kanał niesymetryczny - klient musi znać identyfikator serwera ale nie musi być odwrotnie (ADA, komunikaty systemu QNX).
3. Rozsyłanie komunikatów – komunikaty umieszczane są w globalnej przestrzeni skąd każdy może je pobrać (Linda - przestrzeń krotek)

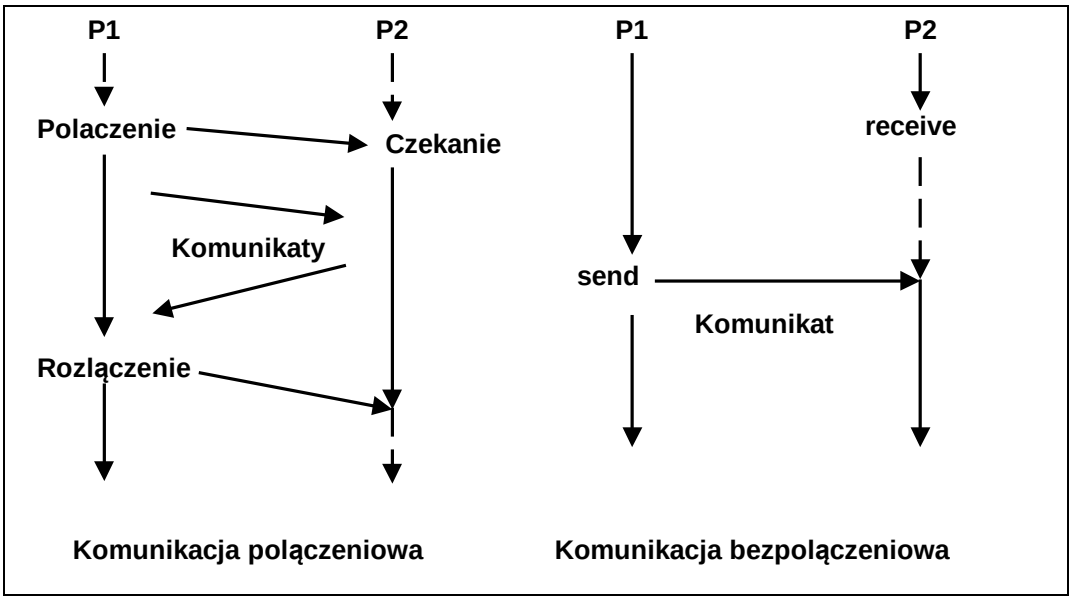
Komunikacja połączeniowa i bezpołączeniowa

Komunikacja połączeniowa

1. Dwa procesy ustanawiają połączenie w którym występuje informacja adresowa
2. Wymieniają dane używając tylko identyfikatora połączenia
3. Rozłączają się

Komunikacja bezpołączeniowa

W każdym przesłaniu występuje pełna informacja adresowa



Rys. 3-3 Komunikacja połączeniowa i bezpołączeniowa

Przeterminowania

Konieczność synchronizowania procesów wymusza konieczność ich blokowania. Procesy są blokowane w oczekiwaniu na pewne zdarzenie. Z różnych powodów (błędy, uszkodzenia) zdarzenie to może nigdy nie nadejść. Spowoduje to trwałą blokadę procesu. Aby temu zaradzić stosuje się przeterminowanie (ang. Timeout) oczekiwania.

Wysłanie komunikatu:
`send(id_odb, bufor_nad, ile, timeout)`

Odbiór komunikatu:
`receive(id_nad, bufor_odb, ile, timeout)`

Gdy proces nie zostanie samoistnie odblokowany po czasie **`timeout`** zrobi to system operacyjny. Funkcja zwróci kod błędu.

Transformacja danych

Odwzorowanie struktur danych w komunikaty

Programy używają rozmaitych struktur danych: struktury, tablice, obiekty, itd. Na poziomie sieci transmitowane są bajty. Nie jest oczywiste jak zamienić je w ciąg bajtów.

Serializacja - zamiana programowych struktur danych w ciąg znaków
Deserializacja- zamiana ciągu znaków na programowe struktury danych

Ujednolicenie typów danych

Wysyłający dane używa specyficznych typów danych jak:
znaki, liczby int, liczby real, stałe boolean, struktury, tablice, teksty różnych formatów, obiekty. W różnych maszynach typy te mogą być reprezentowane w odmienny sposób.

Dwa sposoby reprezentacji liczb

Mniejsze niżej	Intel 80x86, DEC VAX
Mniejsze wyżej	Motorola 68000, Power PC

Sposoby reprezentacji liczb w zależności od typu maszyny

Dla protokołów TCP/IP przyjęto konwencję mniejsze wyżej. Jest to tzw. Format sieciowy.

Przykład 1:
Maszyna M1 - 32 bitowa, system "big endian"
Maszyna M2 - 16 bitowa, system "little endian"

M1 wysyła 32 bitową liczbę x do M2. Aby M2 zinterpretowała liczbę prawidłowo należy:

- Obciąć liczbę x do 16 bitów
- Przestawić bity

Przykład 2:
Maszyna M1 - Tekst ASCII
Maszyna M2 - Tekst Unicode

Możliwe sposoby transformacji danych:

1. Maszyna M1 konwertuje dane do postaci M2 a następnie je wysyła
2. Maszyna M2 konwertuje dane otrzymane z M2 do swojej reprezentacji
3. Przed wysłaniem maszyna M1 konwertuje dane do pewnej zdefiniowanej zewnętrznej reprezentacji (*ang. external reprezentation*). Maszyna M2 po odebraniu konwertuje dane z zewnetrznej reprezentacji do swojej.

Zewnętrzna reprezentacja danych

Zewnętrzna reprezentacja danych (ang. XDR — External Data Representation) zapewnia format pośredni, wykorzystywany przez podczas przesyłania danych pomiędzy systemami klienta i serwera. Wprowadzenie formatu XDR było uwarunkowane koniecznością rozwiązania problemów wymiany danych miedzy systemami heterogenicznymi:

- różne uporządkowanie bajtów w ramach struktur wielobajtowych,
- różne reprezentacje typów danych (jak np. kody EBCDIC i ASCII),
- wyrównywanie struktur.

Format XDR może pośredniczyć w przesyłaniu danych pomiędzy systemami Zapewnia on że system odbierający zawsze otrzyma dane w postaci w jakiej wysłał je proces nadający.

Transformacja postaci danych przy komunikacji nazywa się przetaczaniem danych (*ang. data marshalling*)

Transformacja danych jest konieczna gdy systemy są heterogeniczne

Kroki przetaczania danych:

Przy nadawaniu:
1. Konwersja do zewnętrznej reprezentacji
2. Serializacja danych

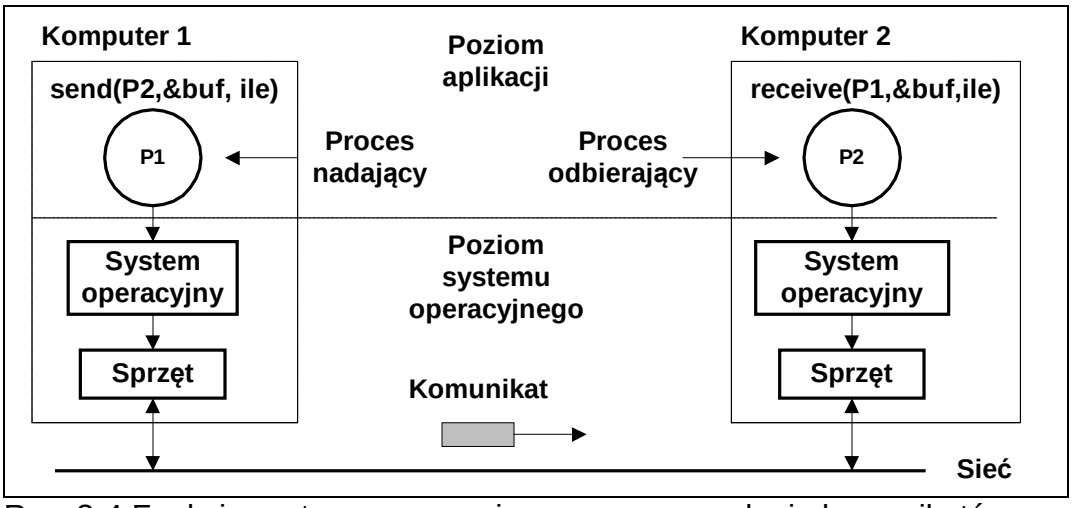
Przy odbiorze:
1. Deserializacja danych
2. Konwersja z zewnętrznej do wewnętrznej reprezentacji

Implementacja przesyłania komunikatów

Przesyłanie komunikatów realizowane jest przez system operacyjny.

Funkcje systemu operacyjnego:

1. Zapewnienie transmisji komunikatu pomiędzy komputerami i ukrycie szczegółów tej transmisji.
2. Wykrywanie i korygowanie błędów transmisji.
3. Składanie i rozkładanie komunikatów w pakiety transmitowane przez sieć.



Rys. 3-4 Funkcja systemu operacyjnego przy przesyłaniu komunikatów

Droga komunikatu od procesu P1 na komputerze 1 do procesu P2 na komputerze 2.