Wprowadzenie do gniazd BSD

Jacek Piotr Nowicki

19 stycznia 2026

Spis treści

1 Wprowadzenie

Z każdym dniem Internet staje się coraz popularniejszy, a to pociąga za sobą fakt, że programy sieciowe nabierają większego znaczenia. Kilka lat temu programowanie aplikacji sieciowych było prawdziwym wyzwaniem. Programista musiał znać wiele szczegółów dotyczących sieci, aby napisać prostą aplikację działającą w Internecie. Dzisiaj sytuacja wygląda zupełnie inaczej. Istnieją wygodne narzędzia do programowania sieciowego, których możliwości zależą od systemu operacyjnego i języka programowania. Jednym z nich jest interfejs gniazdowy, zwany też gniazdami BSD (ang. Berkeley Software Distribution), wywodzący się z systemu operacyjnego BSD utworzonego na Uniwersytecie w Berkeley. Interfejs gniazd pozwala programiście skoncentrować się na tworzeniu aplikacji, ukrywając trudności związane z programowaniem sieciowym na niskim poziomie.

Celem niniejszej pracy jest przedstawienie teoretycznych i praktycznych podstaw zagadnień dotyczących programowania sieciowego, wykorzystującego interfejs gniazd BSD. Praca składa się z pięciu rozdziałów i jest podzielona na dwie części :

- Pierwsze dwa rozdziały prezentują teoretyczny aspekt tematu. Opisałem w nich historię powstania gniazd BSD (rozdział 1) oraz podstawowe definicje z nimi związane (rozdział 2).
- Kolejne trzy rozdziały składają się na część praktyczną pracy. Przedstawiłem problematykę adresowania gniazd (rozdział 3) i funkcje systemowe wchodzące w skład interfejsu gniazd (rozdział 4). Zaawansowane zagadnienia związane z gniazdami opisałem w rozdziale 5. W tej części pracy umieściłem kilkanaście przykładowych programów, które będą pomocne w zrozumieniu przedstawionej teorii.

Wszystkie przykładowe programy napisałem w języku ANSI C i przetestowałem na komputerze z zainstalowanym systemem LINUX Red Hat 7.3 (Valhalla). Programy te umieściłem na dyskietce dołączonej do pracy.

1.1 Krótki rys historyczny

4 października 1957 roku Związek Radziecki umieścił na orbicie okołoziemskiej pierwszego sztucznego satelitę o nazwie “Sputnik”, bijąc tym samym Amerykanów w dziedzinie podboju kosmosu. W roku 1962 wybuchł kryzys na Kubie. Te wydarzenia spowodowały, że Stany Zjednoczone zaczęły zabezpieczać się przed ewentualną wojną atomową. Rząd amerykański nie ograniczając wydatków na rozwój nowoczesnych technologii powołał do życia agencję

ARPA (Advanced Research Project Agency – Agencję do Zaawansowanych Projektów Technicznych), której celem było stworzenie zdecentralizowanego systemu komunikacji (tak jakby bez punktu centralnego), który działałby nawet po częściowym zniszczeniu w ataku nuklearnym części podłączonych do niego urządzeń. Owocem pracy agencji było powstanie w 1969 roku sieci ARPAnet - pierwszej rozległej sieci stworzonej na potrzeby ARPA. Łączyła centra badań i uczelnie, umożliwiając im pracę nad technologiami sieciowymi. Po pewnym czasie ARPA została przekształcona w agencję DARPA (ang. Defense Advanced Research Projects Agency – Agencja do spraw Wojskowych Zaawansowanych Projektów Technicznych).

Równocześnie z rozwojem ARPAnetu, od roku 1969 rozpoczął się rozwój systemu UNIX. Na Uniwersytecie w Berkeley (ang. University of California, Berkeley, w skrócie UCB) utworzono własną odmianę systemu UNIX znaną jako BSD. Rozwijany przez studentów i naukowców system stawał się coraz bardziej funkcjonalny. W miarę upływu czasu pojawiały się jego kolejne wersje, których możliwości i udogodnienia wzrastały w zaskakującym tempie. W efekcie tych prac powstał system 3BSD będący następcą wersji 1BSD i 2BSD. Ten niesamowity rozwój zwrócił uwagę agencji DARPA, która postanowiła dofinansowywać dalsze badania związane z tym systemem. W 1977 roku UCB i DARPA nawiązały współpracę, której celem było stworzenie systemu operacyjnego zdolnego pracować na licznych platformach sprzętowych używanych przez agencję. Specjalna grupa – Grupa Badań nad Systemami Operacyjnymi (ang. CSRG - Computer System Research Group) powołana przez UCB miała na celu dostosowanie systemu BSD do potrzeb agencji. W 1980 roku światło dzienne ujrzał system 4BSD, a następnie 4.1BSD ulepszony pod kątem wydajności. Jednak ARPA na tym nie poprzestała i przedłużyła kontrakt zlecając UCB włączenie do systemu obsługę sieci ARPAnet i poprawienie komunikacji międzyprocesowej. Dalekowzroczni programiści "poszli dalej" i włączyli do systemu obsługę innych protokołów sieciowych. Efektem ich pracy było powstanie w 1983 roku systemu 4.2BSD, wyposażonego w solidne oprogramowanie sieciowe, w którego skład wchodził między innymi interfejs gniazd BSD. W roku 1990 wprowadzono kilka zmian do interfejsu gniazdowego w związku z ukazaniem się systemu 4.3BSD Reno, w którym oprogramowanie protokołów OSI weszło do jądra systemu.

Punktem wyjścia dla wielu wersji systemu UNIX był kod oprogramowania sieciowego jakieś wersji systemu BSD, zawierającego oprogramowanie interfejsu gniazdowego.

1.2 Wprowadzenie do gniazd

1.2.1 Komputer w sieci

System komputerowy ma charakter wspólnoty symbiotycznej – sprzęt i oprogramowanie zależą ściśle od siebie nawzajem i nie mogą działać osobno. Sprzęt składa się z urządzeń peryferyjnych, procesorów, pamięci, dysków i innych urządzeń elektronicznych, które razem stanowią całość komputera. Jednak komputer bez oprogramowania jest bezużyteczny. Oprogramowanie sterujące, niezbędne do działania komputera, nazywamy systemem operacyjnym. Jest to niskopoziomowe oprogramowanie obsługujące sprzęt i zapewniające określony zestaw usług programom użytkowym.

Popularnym systemem operacyjnym, który może pracować w sieci jest system UNIX lub jego nowoczesna implementacja LINUX – wielozoadaniowy i wielodostępowy system operacyjny klasy UNIX, dystrybuowany na podstawie licencji GNU (ang. GNU General Public License) opracowanej przez Free Software Foundation. Pomyślany początkowo jako narzędzie dla hobbystów i profesjonalnych programistów, LINUX staje się w coraz większym stopniu systemem operacyjnym dla "przeciętnego użytkownika". Będąc zarazem systemem wydajnym, szybkim i darmowym, zdobywa domowe i profesjonalne środowiska komputerowe.

Ściśle związany z UNIX'em (LINUX'em) jest język C, będący językiem ogólnego stosowania. Charakteryzuje się prostotą wyrażeń, nowoczesnym sterowaniem, nowoczesnymi strukturami danych oraz bogatym zestawem operatorów. Język ten opracował i zrealizował Dennis Ritchie dla systemu operacyjnego UNIX działającego na mikrokomputerze DEC PDP-11.

Dysponując komputerem z zainstalowanym systemem operacyjnym (na przykład systemem LINUX), kompilatorem języka C oraz dowolnym edytorem tekstu (na przykład emacs) możemy napisać program. Program jest to plik wykonywalny, utworzony zazwyczaj przy użyciu programu łączącego i znajdujący się w pamięci dyskowej. Program staje się procesem wtedy, kiedy jest wykonywany przez system operacyjny. Często się mówi, że program jest obiektem statycznym, czyli formalnym opisem tego, co ma być wykonane, a proces jest obiektem dynamicznym, czyli ciągiem sekwencyjnie wykonywanych przez system operacyjny instrukcji programu. W jądrze UNIX'a istnieje pewna liczba (zazwyczaj ograniczona) tak zwanych funkcji systemowych (ang. shell call), za pomocą których aktywny proces może uzyskać różne usługi ze strony jądra systemu. Większość funkcji systemowych zwraca wartość

całkowitą dodatnią w przypadku pomyślnego wykonania. Jeżeli podczas wykonywania funkcji wystąpi błąd, to funkcja zazwyczaj zwraca wartość -1 , a zmienna globalna errno otrzyma wartość całkowitą dodatnią, wskazującą na rodzaj błędu. Wszystkim dodatnim wartościom zmiennej errno odpowiadają stałe, których nazwy składają się z wielkich liter, zaczynają się od litery E i są zdefiniowane w pliku nagłówkowym . Żaden błąd nie ma wartości 0.

1.2.2 Sieć komputerowa

Sieć komputerowa jest systemem komunikacyjnym łączącym systemy końcowe, zwane stacjami sieciowymi (ang. host computer) . Komputery są połączone jakimś medium fizycznym, na przykład kablem komunikacyjnych. Niezależnie od sposobu połączenia komputerów ze sobą, celem sieci jest zapewnienie komunikacji między poszczególnymi stacjami – komputerami do niej podłączonymi.

Najpopularniejszą, siecią komputerową jest Internet – największa sieć, łącząca sieci komputerowe na całym świecie ("sieć sieci"). Składa się z setek tysięcy kilometrów światłowodów, łącz satelitarnych oraz całej masy mniej lub bardziej skomplikowanym urządzeń. Użytkownicy Internetu, których liczbę szacuje się w milionach, mogą za jego pośrednictwem mieć dostęp do "oceanu" informacji, komunikować się między sobą, wymieniać dane itp.

Większość programów użytkowych, które są przeznaczone do działania w sieci komputerowej, można podzielić na dwie grupy. Pierwsza z nich to aplikacje zwane serwerami, które dostarczają usług w Internecie, drugi zbiór składa się z programów zwanych klientami, które z tych korzystają. Termin serwer powoduje czasem nieporozumienia. Formalnie oznacza on program, który czeka biernie na wykonanie pewnej usługi, a nie komputer, który ją wykonuje. Jeśli jednak jakiś komputer jest wyznaczony do wykonywania jednego lub więcej serwerów, to sam jest czasami (niepoprawnie) nazywany serwerem. Producenci sprzętu komputerowego pogłębiają jeszcze te nieporozumienia, gdyż nazywają serwerami komputery o dostatecznie szybkim procesorze, odpowiednio pojemnej pamięci i stosownie wyrafinowanym systemie operacyjnym.

Komunikacja w sieci oparta jest na protokołach. Protokoły są to formalne reguły postępowania. Na przykład w stosunkach międzynarodowych, protokoły minimalizują problemy wynikające z różnic kulturowych przy współpracy różnych narodów. Zgadzając się na wspólny zestaw ogólnie przyjętych reguł, które są niezależne od jakichkolwiek zwyczajów narodowych, protokoły dyplomatyczne minimalizują nieporozumienia; każdy wie jak się zachować

i jak interpretować zachowania innych. Podobnie przy łączności komputerów konieczne jest zdefiniowanie reguł, które rządzą komunikacją w sieci. Ponieważ zdefiniowanie takich zasad jest niesłychanie trudne, Międzynarodowa Organizacja ds. Standardów (ang. International Standards Organization [ISO]) opracowała model architektury służący do opisu zasad komunikacji w sieci. Model odniesienia łączenia systemów otwartych (ang. Open System Interconnect [OSI] Reference Model) gwarantuje uzyskanie wspólnego mianownika dla zagadnień komunikacyjnych. Wszystko co zostało zdefiniowane za pomocą tego modelu jest ogólnie rozumiane i szeroko stosowane przez osoby zajmujące się komunikacją w sieci. Model OSI składa się z siedmiu warstw na których znajduje się jeden lub więcej protokołów, które opisują sposoby realizacji zadań przeznaczonych dla danej warstwy. Zbiór protokołów obowiązujących na różnych warstwach i mogących stanowić podstawę dla użytecznej sieci nazywamy rodziną protokołów (ang. protocol family).

Rodzinę TCP/IP (ang. Transmission Control Protocol / Internet Protocol) będącą rodziną protokołów odpowiedzialną za komunikację w Internecie również możemy przedstawić za pomocą modelu OSI, ale za pomocą czterech warstw : warstwa kanałowa; warstwa sieciowa; warstwa transportowa; warstwa zastosowań. W modelu czterowarstwowym rodzinę protokołów wyznaczają dwie warstwy : transportowa (protokół TCP i protokół UDP) oraz sieciowa (protokół IP). Natomiast w jedną warstwę kanałową są połączone protokoły i cechy charakterystyczne sieci na poziomie jej topologii oraz użytego sprzętu (Ethernet albo Token ring). Przestrzenią programów użytkowych jest zaś warstwa zastosowań.

Do najważniejszych protokołów z rodziny TCP/IP należą :

- Protokół IP (ang. Internet Protocol) jest protokołem warstwy sieciowej i jest on fundamentalnym elementem Internetu. Do najważniejszych jego zadań należą między innymi obsługa doręczania pakietów dla protokołów z warstwy transportowej takich jak TCP i UDP oraz definiowanie schematu adresowania w Internecie.

- Protokół TCP (ang. Transmission Control Protocol) jest protokołem połączeniowym znajdującym się na warstwie transportowej. Umożliwia niezawodne i w pełni dwukierunkowe (ang. full-duplex) przesyłanie strumienia danych. W większości internetowych programów stosuje się ten protokół.

- Protokół UDP (ang. User Datagram Protocol) jest protokołem bezpołączeniowym znajdującym się na warstwie transportowej. W odróżnie-

niu od protokołu TCP, który jest niezawodny, protokół UDP nie daje gwarancji, że dana wiadomość dotrze do wyznaczonego celu.

Aby komputery podłączone do Internetu mogły się ze sobą komunikować, musi istnieć pewien sposób ich identyfikacji. Innymi słowy, każda stacja podłączona do sieci musi mieć przypisany numer identyfikacyjny, który jest niepowtarzalny. Typowy adres stacji składa się z identyfikatora sieci oraz identyfikatora stacji w tej sieci. W Internecie komputery są identyfikowane za pomocą 32-bitowej liczby całkowitej, znanej jako adres IP. Adresy te są wyznaczane przez upoważniony do tego węzeł centralny – Sieciowe Centrum Informacyjne czyli NIC (ang. Network Information Center), zarządzane przez firmę SRI International. Aby adresy IP były łatwo czytelne, zostały podzielone na 8-bitowe liczby zwane oktetami (format ten często jest nazywany kropkową notacją czwórkową). Ponieważ łatwiej zapamiętać nazwy niż liczby, usługa nazewnictwa domen (ang. Domain Name Service – DNS) przyporządkowuje unikatowym adresom liczbowym IP nazwy. Na przykład komputer o nazwie quark.physics.grouch.edu ma adres IP 0x954C0C04, zapisywany jako 146.76.12.4. Nie wszystkie adresy sieci i komputerów są dostępne dla użytkowników. Na przykład adresy, których pierwszy bajt jest większy od 223 są zarezerwowane. Nas będzie najbardziej interesował adres 127.0.0.1 będący adresem sieci zwrotnej (ang. loopback adress). Sieć zwrotna składa się z jednego komputera o nazwie localhost i adresie 127.0.0.1. Oznacza to, że samotny komputer (nie podłączony do żadnej sieci) z zainstalowanym systemem UNIX działa w sieci jednoelementowej, którą stanowi ten pojedyczy komputer. W przykładach opisanych w dalszej części pracy użyłem właśnie sieci zwrotnej. Adres takiego komputera można znaleźć w pliku hostów sieciowych /etc/hosts.

Komputer w sieci może działać jako stacja jednosieciowa lub wielosieciowa, czyli stacja połączona z co najmniej dwiema sieciami. Każda sieć, z którą porozumiewa się komputer, posiada przypisany jej interfejs sprzętowy, przez który następuje dostęp do sprzętu sieciowego. Komputer może mieć odmienne nazwy w każdej z sieci, a z pewnością będzie miał odrębne adresy. Wypływa z tego wniosek, że każdy adres w Internecie określa w jednoznaczny sposób daną stację, ale nie każda stacja musi mieć dokładnie jeden adres.

Znajomość adresu IP danego komputera, czyli umiejętność zlokalizowania go w sieci nie wystarcza do nawiązania komunikacji, ponieważ na komputerze o danym adresie IP może działać wiele aplikacji pełniących rolę serwera. W celu rozróżnienia tych procesów używa się 16-bitowych numerów portów. Zarówno dla protokołów TCP jak i UDP zdefiniowano grupę por-

tów ogólnie znanych (ang. well known ports), przeznaczonych do wykonywania ogólnie znanych usług. Na przykład, w każdej implementacji rodziny protokołów TCP/IP, która pozwala korzystać z protokołu FTP (ang. File Transfer Protocol), serwerowi FTP przypisano ogólnie znany port o numerze 21. Protokół TFTP, czyli prymitywny protokół przesyłania plików (ang. Trivial File Transfer Protocol), ma w protokole UDP przyporządkowany ogólnie znany port 69. Klienci łącząc się z serwerem używają portów efemerycznych (ang. ephemeral port), czyli krótkotrwałych. Numery tych portów są zazwyczaj automatycznie wyznaczane klientom przez protokoły TCP i UDP, które gwarantują, że nie ma portu przypisanego do dwóch procesów jednocześnie i że numer takiego portu jest zawsze większy od numerów dobrze znanych portów. Dokument RFC 1700 zawiera wykaz numerów portów wyznaczonych przez organizację IANA (ang. Internet Assigned Numbers Authority). Każdy numer portu należy do jednego z trzech zakresów:

- Porty ogólnie znane (ang. well-known ports) mają numery z przedziału 01023. Nadzór nad tymi numerami portów i ich przyporządkowanie należy do organizacji IANA. W miarę możności ten sam numer portu jest przypisany do tej samej usługi w obu protokołach – TCP i UDP. Na przykład port 80 przyporządkowano do serwera sieci WWW dla obu protokołów, pomimo że obecnie we wszystkich implementacjach używa się tylko protokołu TCP.

- Porty zarejestrowane (ang. registered ports) mają numery z przedziału 102449151. Organizacja IANA nie sprawuje nad nimi nadzoru, lecz dla wygody użytkowników sieci rejestruje numery portów i sporządza wykazy ich przyporządkowania. W miarę możności ten sam numer portu jest przypisany do tej samej usługi w obu protokołach – TCP i UDP. Na przykład porty od 6000 do 6063 przyporządkowano do serwera X-Windows dla obu protokołów.

- Porty dynamiczne lub prywatne mają numery z przedziału 49152-65535. Organizacja IANA nic nie mówi o tych portach. Są to porty nazywane portami efemerycznymi.

W systemie LINUX, informacja o tym, jakie usługi odpowiadają jakim portom przechowywana jest w pliku /etc/services. Do najbardziej znanych usług należą :

- port o numerze 7 - usługa echo.

- port o numerze 9 - usługa discard.

- port o numerze 13 - usługa daytime.
- port o numerze 19 - usługa chargen.
- port o numerze 21 - usługa FTP.
- port o numerze 23 - usługa Telnet.
- port o numerze 25 - serwer pocztowy używający protokołu SMTP.
- port o numerze 37 - usługa time.
- port o numerze 53 - usługa DNS.
- port o numerze 80 - serwer WWW.

1.2.3 Model komunikacji klient-serwer

Standardowym modelem komunikacji w sieci jest model "klient-serwer". Polega on na tym, że jeden proces zwany serwerem dostarcza usługi pod określonym adresem IP i numerem portu. Drugi proces zwany klientem znając adres IP i numer portu serwera, może skorzystać z jego usługi. Przykładowy scenariusz mógłby wyglądać tak :

- Proces, zwany serwerem, rozpoczyna pracę w pewnym systemie komputerowym. Po zainicjowaniu pracy serwer "zasypia" czekając na proces, zwany klientem, który skontaktuje się z nim, zamawiając jakąś usługę.
- Proces zwany klientem, rozpoczyna pracę albo w tym samym systemie, co serwer, albo w innym systemie połączonym z systemem serwera za pomocą sieci komputerowej. Klient wysyła zamówienie do serwera, prosząc o pewien rodzaj usługi (na przykład serwer musi podać klientowi dokładną godzinę).
- Kiedy serwer zakończy wykonywanie usługi dla klienta, wtedy znowu zasypia i oczekuje na nadejście następnego żądania.

1.2.4 Definicja gniazda

Kombinacja adresu IP i numeru portu nosi nazwę gniazda (ang. socket). Można powiedzieć, że gniazdo jest programową abstrakcją używaną do reprezentowania końcówki połączenia między dwoma komputerami. Dla każdego połączenia istnieją gniazda na obydwu uczestniczących w nim komputerach (aplikacjach działających na tych komputerach). Wyobraźmy sobie rozciągający się pomiędzy komputerami kabel, którego końce są wpięte do tych

gniazd. Formalnie gniazdo powinniśmy zdefiniować jako zbiór trzyelementowy :

- protokół komunikacyjny, adres-obcy, proces-obcy - pierwsze gniazdo
- protokół komunikacyjny, adres-lokalny, proces-lokalny – drugie gniazdo

Wartością adres-lokalny i adres-obcy są identyfikatory stacji lokalnej oraz stacji odległej. Natomiast elementy proces-lokalny i proces-obcy określają konkretne procesy działające na obu komputerach. Te trzyelementowe zbiory stanowią punkty końcowe komunikacji w sieci zwane półasocjacjami.

Jeżeli rozpocznie się komunikacja między powyższymi komputerami, zostanie utworzony pięcioelementowy zbiór wystarczający do pełnego określenia obu komunikujących się ze sobą procesów.

- protokół, adres-lokalny, proces-lokalny, adres-obcy, proces, obcy

Ten zbiór nazywamy asocjacją.

Interfejs wykorzystywany przez program użytkowy przy interakcji z oprogramowaniem protokołów warstwy transportowej nazywa się interfejsem programu użytkowego (ang. Application Program Interface – API). Interfejs API określa zestaw operacji, które program użytkowy może wykonywać w ramach interakcji z oprogramowaniem protokołów. Większość systemów oprogramowania definiuje interfejs API, podając zestaw funkcji, które program może wywoływać, oraz argumentów, których każda z nich się spodziewa. Zwykle API zawiera oddzielną funkcję dla każdej operacji podstawowej. Interfejs API może na przykład zawierać funkcję, która jest wykorzystywana do ustanawiania komunikacji, oraz inną funkcję, która jest używana do wysyłania danych.

Standardy protokołów komunikacyjnych nie określają zwykle interfejsu, którego programy mają używać przy interakcji z nimi. Protokoły określają ogólne operacje, które powinny być udostępnione, oraz pozwalają systemowi operacyjnemu na zdefiniowanie konkretnego interfejsu API, którego programy będą używać do wykonywania tych operacji. Chociaż standardy protokołów pozwalają projektantom systemów operacyjnych na wybranie interfejsu API, to wielu z nich wybrało interfejs gniazd BSD, który stał się standardem interfejsów API.

1.2.5 Atrybuty gniazd

Znając definicje gniazd można zrobić krok naprzód i zająć się podstawowymi parametrami, które mają wpływ na nowo utworzone gniazdo. Gniazda mu-

szą mieć swoją domenę, czyli określone środowisko, w którym będą działać. Powinny mieć typ, czyli sposób komunikowania się w sieci. Zazwyczaj do pełnego zdefiniowania gniazda wystarczają powyższe atrybuty. Czasem zdarzają się sytuację, gdy programista musi ręcznie przypisać gniazdu protokół komunikacyjny. Gniazda mają również swój adres, który jest kojarzony z ich nazwą

1.2.6 Rodzaj transmisji

Rodzaj transmisji informuje nas w jaki sposób gniazda będą traktować transmitowane dane. Gniazdo może przyjąć jedną z poniższych transmisji :

Transmisja połączeniowa

- Transmisja połączeniowa gwarantuje dostarczenie wszystkich pakietów danych w takiej kolejności, w jakiej zostały wysłane. Najpierw jest ustanawiane łącze pomiędzy dwoma komunikującymi się procesami, a dopiero potem zachodzi wymiana danych. Rozwiązanie takie oznacza, że pomiędzy tymi procesami jest wyznaczona trasa oraz że obydwaj uczestnicy transmisji są aktywni. Ustanowienie kanału komunikacyjnego wymaga dodatkowego czasu. Ponadto większość protokołów połączeniowych gwarantuje dostarczenie wiadomości, co jeszcze bardziej wydłuża transmisję, gdyż zachodzi konieczność przeprowadzania obliczeń i weryfikowania poprawności. Protokoły połączeniowe są niezawodne, ale niestety ta niezawodność jest uzyskana kosztem czasu. Transmisja połączeniowa przypomina rozmowę telefoniczną. Najpierw ustanawia się połączenie z rozmówcą, następnie przez pewien czas wymienia się dane i wreszcie zamyka się połączenie.Rozważmy typowy scenariusz transmisji połączeniowej. Serwer jest procesem oczekującym na określoną liczbę połączeń klienta. Jego zadaniem jest obsługa żądań klientów. Serwer musi oczekiwać połączeń pod powszechnie znaną nazwą. Na przykład w protokole TCP/IP nazwą tą będzie adres IP oraz numer portu. Najpierw aplikacja serwera tworzy gniazdo, które jest po prostu zasobem systemu operacyjnego przypisanym do procesu serwera. Służy do tego funkcja systemowa socket(). Następnie należy związać to gniazdo z jego powszechnie znaną nazwą (adresem zdefiniowanym w gniazdowej strukturze adresowej). Zadanie to jest realizowane za pomocą funkcji bind(). Teraz należy przełączyć gniazdo w tryb nasłuchiwania, co umożliwia wywołanie systemowe listen(). Na zakończenie, gdy klient próbuje nawiązać łączność, serwer może zaakceptować to połączenie za pomocą funkcji accept(), która tworzy nowe gniazdo niezależne od

gniazda nazwanego i przeznaczone do komunikacji z danym klientem. Od strony klienta sytuacja jest o wiele prostsza, gdyż nawiązanie połączenia wymaga mniejszej liczby kroków. Aplikacja klienta tworzy gniazdo nienazwane używając funkcji socket(). Następnie próbuje nawiązać połączenie z serwerem, wykorzystując jego nazwane gniazdo jako adres. Realizuje to używając wywołania systemowego connect() . Po nawiązaniu połączenia oba gniazda za pomocą funkcji systemowych read() i write() zapewniają dwukierunkową komunikację.

Transmisja bezpołączeniowa

- Transmisja bezpołączeniowa nie gwarantuje dostarczenia danych ani przekazania ich w odpowiedniej kolejności. Pakiety mogą zostać zgubione lub pomieszane w czasie transmisji w związku z błędami sieci lub z innych powodów. Transmisja bezpołączeniowa przypomina usługi pocztowe. Każda przesyłka zawiera pełny adres odbiorcy. Przesyłki wysyłane pod ten sam adres mogą przebywać różne trasy, zanim zostaną doręczone. Dlatego istnieje możliwość, że dwa listy nadane pod ten sam adres w krótkim odstępie czasu, dotrą do adresata w odwrotnej kolejności. Nie ma też pełnej gwarancji, że przesyłka będzie pomyślnie dostarczona. Programista implementujący gniazda oparte na transmisji bezpołączeniowej musi sam zabezpieczyć swój program przed skutkami ich zawodności. Transmisja bezpołączeniowa jest wielokrotnie szybsza niż połączeniowa. Może być wykorzystywana do takich aplikacji, w których nie ma większego znaczenia to, iż kilka pakietów danych zostanie tu i tam zagubionych, ponieważ szybkość jest dla nich najważniejsza. Przykładem takiej aplikacji może być gra sieciowa przeznaczona dla wielu użytkowników. Każdy gracz stosuje datagramy, aby wysłać do innych graczy informacje o swojej aktualnej pozycji w grze. Jeżeli nawet któryś z pakietów nie dotrze do któregoś z graczy, bardzo szybko otrzyma on następny pakiet, dzięki czemu będzie miał wrażenie płynności gry.

Oto przykład typowego scenariusza transmisji bezpołączeniowej. Aplikacja serwera tworzy gniazdo nienazwane za pomocą funkcji socket(). Następnie nadaje mu nazwę używając wywołania systemowego bind(). Odmienność polega na tym, że proces serwera nie nasłuchuje ani też nie akceptuje połączeń. Zamiast tego aplikacja serwera po prostu oczekuje na nadchodzące dane. Serwer może odebrać dane używając funkcji recvfrom(). Podobnie klient nie ustanawia połączenia z serwerem, ale po prostu wysyła do niego datagram za pośrednictwem wywołania systemowego sendto().

1.2.7 Domena gniazd

Domena gniazd określa środowisko sieciowe, z którego będą korzystać gniazda oraz sposób zapisu adresów gniazd identyfikujących jeden z końców połączenia komunikacyjnego. Gniazda mogą występować w następujących środowiskach:

- Gniazda w dziedzinie UNIX'a Z gniazd w dziedzinie UNIX'a (ang. Unix domain protocols) możemy korzystać do komunikowania się z procesami tylko w obrębie tego samego systemu operacyjnego. W tej dziedzinie można implementować zarówno gniazda połączeniowe jak i bezpołączeniowe. Można przyjąć, że obie implementacje są niezawodne, ponieważ znajdują się wewnątrz jądra systemu i nie są przesyłane przez urządzenia zewnętrzne, takie jak linie komunikacyjne łączące komputery. Nie są potrzebne sumy kontrolne ani inne środki zapewniające niezawodność. Protokołem obsługującym gniazda połączeniowe jest unixstr, a gniazda bezpołączeniowe – unixdg. Adresem gniazda jest nazwa pliku przechowywana w systemie plików.

- Gniazda w dziedzinie Internetu Gniazd z domeny Internetu możemy używać do komunikacji między procesami znajdującymi się na różnych komputerach podłączonych do sieci. W tym środowisku możemy implementować zarówno gniazda połączeniowe jak i bezpołączeniowe. Używanym protokołem dla gniazd połączeniowych jest TCP (ang. Transmission Control Protocol), a dla gniazd bezpołączeniowych – UDP (ang. User Datagram Protocol). Oba protokoły działają na bazie protokołu niższego poziomu IP (ang. Internet Protocol). Gniazda połączeniowe zapewniają niezawodne dostarczanie danych dzięki pełnodupleksowemu połączeniu implementowanemu za pomocą datagramów IP. Niezawodność uzyskuje się przez użycie mechanizmu potwierdzeń i retransmisji. Jeżeli nadawca nie dostanie na czas potwierdzenia odebrania pakietu przez odbiorcę, to zakłada, że dane zginęły i retransmituje je. Po pewnej liczbie ponownych retransmisji oprogramowanie TCP zaniecha wysyłania danych, przy czym cały czas przeznaczony na próby wysyłania danych wynosi od 4 do 10 minut (zależnie od implementacji systemu). Gniazda bezpołączeniowe są zawodne, więc jeżeli chcemy uzyskać pewność, że datagram dojdzie do odbiorcy, to musimy nasze oprogramowanie użytkowe uzupełnić o wiele właściwości takich jak potwierdzenie uzyskania danych przez odbiorcę, obsługę czasu oczekiwania na odpowiedź, ponawianie retransmisji itp. Adres gniazda internetowego zajmuje 32 bity i składa się z dwóch części : identyfikatora sieci oraz identyfikatora stacji w tej sieci. Oprócz tego używa

się 16-bitowego numeru portu służącego do identyfikacji konkretnego procesu na komputerze o danym adresie IP.

1.2.8 Protokół kominikacyjny

Czasem zdarza się, że mechanizm transportowy potrafi korzystać z różnych protokołów komunikacyjnych, aby określić żądany typ gniazda. Wtedy programista może wskazać najwłaściwszy protokół dla danego gniazda. Zaleca się wyrażenie zgody, aby system operacyjny wybrał domyślny protokół komunikacyjny.