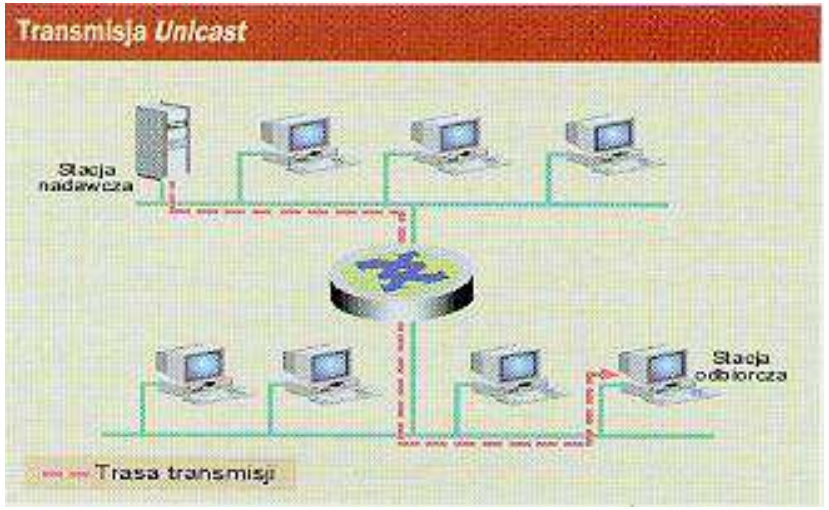
Typy transmisji sieciowych

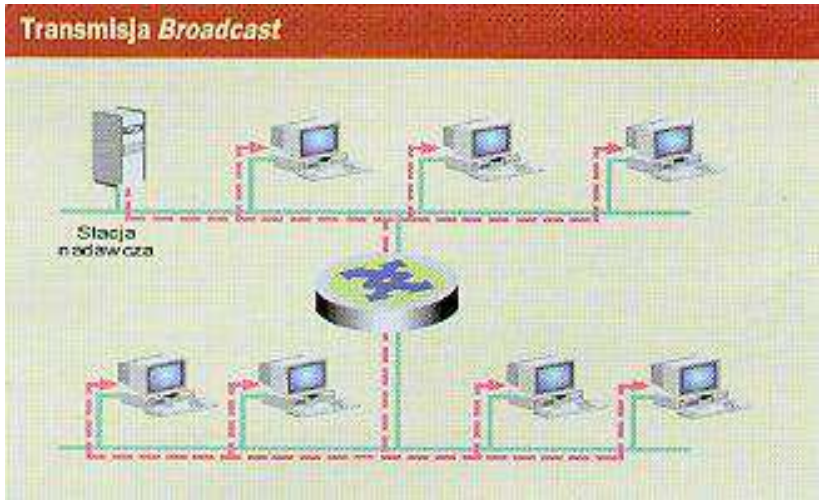
Komunikacja **jednokanałowa** (*point-to-point, unicast*)

np. połączenie telefoniczne drutowe (aparat końcowy do centrali)



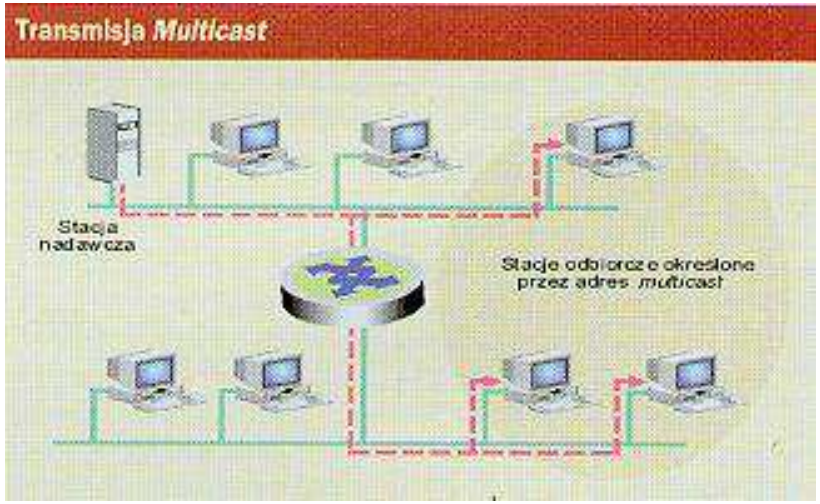
Komunikacja typu **rozgłaszania** (*broadcast*)

np. transmisja radiowa, satelitarna, itp.



Typy transmisji sieciowych — multicast

Podobny do *broadcast*u **multicast** oznacza jednoczesne nadawanie do grupy odbiorców. Należy odróżnić transmisję komunikatu *multicast* do określonej grupy (pojedynczy strumień danych) od transmisji jednego komunikatu wiele razy do grupy (zduplikowany strumień danych).



Technologie *multicast* nie są nowym wynalazkiem, ale dotychczas były mało popularne i rozwijane ze względu na komplikacje w niezbędnych technologiach i standardach. Np. każdy router, nawet w małej sieci domowej lub osiedlowej, powinien być przystosowany do odebrania transmisji *multicast*, zbadania adresu grupy *multicast*, i zadecydowania, czy transmisję należy przekazywać do wnętrza sieci, i konkretnie do których jej części.

W kontekście nabierających popularności szerokopasmowych transmisji wideo te technologie jednak stają się coraz ważniejsze. Zamiast dublować strumień wideo, jak również wysyłać do wszystkich sieci (na świecie) lepiej kierować go do zdefiniowanej grupy.

Typy komunikacji — komunikacja połączeniowa

Wygodnie jest rozważać dwa zasadniczo różne modele komunikacji: połączeniowy i bezpołączeniowy.

Komunikacja połączeniowa jest oparta na zbudowaniu połączenia między stronami, połączenie inicjuje jedna strona, ale utrzymują je obie. Dopóki połączenie istnieje, każda ze stron może nadawać w dowolnym momencie, a druga strona odbiera tę transmisję. Po rozłączeniu, dalsza komunikacja jest niemożliwa do czasu ponownego nawiązania połączenia.

Dobrą analogią komunikacji połączeniowej jest rozmowa przez telefon. Strona inicjująca połączenie musi znać numer telefonu (adres) strony przyjmującej. Strona przyjmująca może nie mieć świadomości numeru dzwoniącego. (Czasami technologia sieci pozwala odbiorcy poznać ten numer, tzw. Caller-ID, ale nie jest to potrzebne do komunikacji.) Jednak po nawiązaniu połączenia żadna ze stron nie musi już pamiętać numeru telefonu drugiej strony.

Typowo w komunikacji połączeniowej strumień danych dochodzi w tej samej postaci w jakiej został nadany (nie ma zamiany kolejności), aczkolwiek przy zawodnym medium jest możliwe przekłamanie, albo utrata części danych.

Typy komunikacji — komunikacja bezpołączeniowa

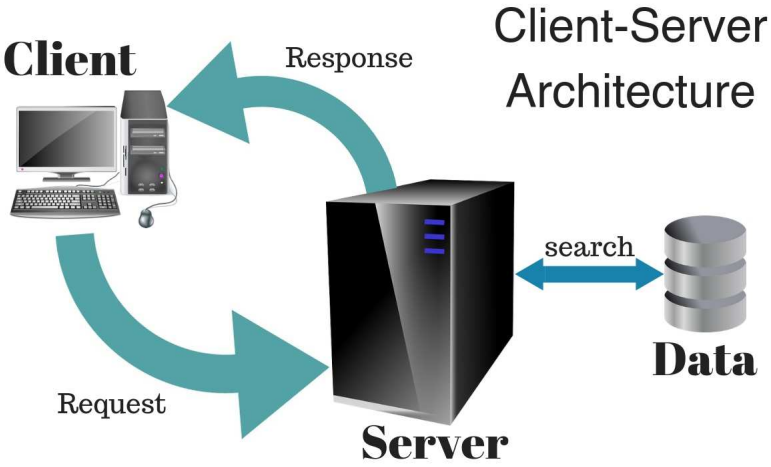
Komunikacja bezpołączeniowa jest oparta na wysyłaniu w pełni adresowanych pakietów danych, z których każdy może być niezależnie doręczony odbiorcy. W każdej chwili możemy wysłać odbiorcy pakiet danych, pod warunkiem, że znamy jego adres.

Podobną analogią komunikacji bezpołączeniowej jest korespondencja listowa. Aby wysłać komuś list trzeba znać jego adres, i żeby odbiorca mógł odpowiedzieć musi on znać adres nadawcy. Przesyłka może być doręczona z adresem nadawcy lub bez tego adresu. (Tradycyjna poczta nie oferuje usługi dostarczenia wraz z listem adresu nadawcy, ale w sieciowej komunikacji bezpołączeniowej takie możliwości zwykle istnieją.)

Typowo w komunikacji bezpołączeniowej możliwa jest zamiana kolejności niektórych pakietów (ich doręczenie w innej kolejności niż były nadane), bo trudno jest kontrolować media komunikacyjne aby tej kolejności przestrzegały. Przekłamania i gubienie przesyłek jest możliwe podobnie jak w komunikacji połączeniowej.

Modele komunikacji: klient-serwer

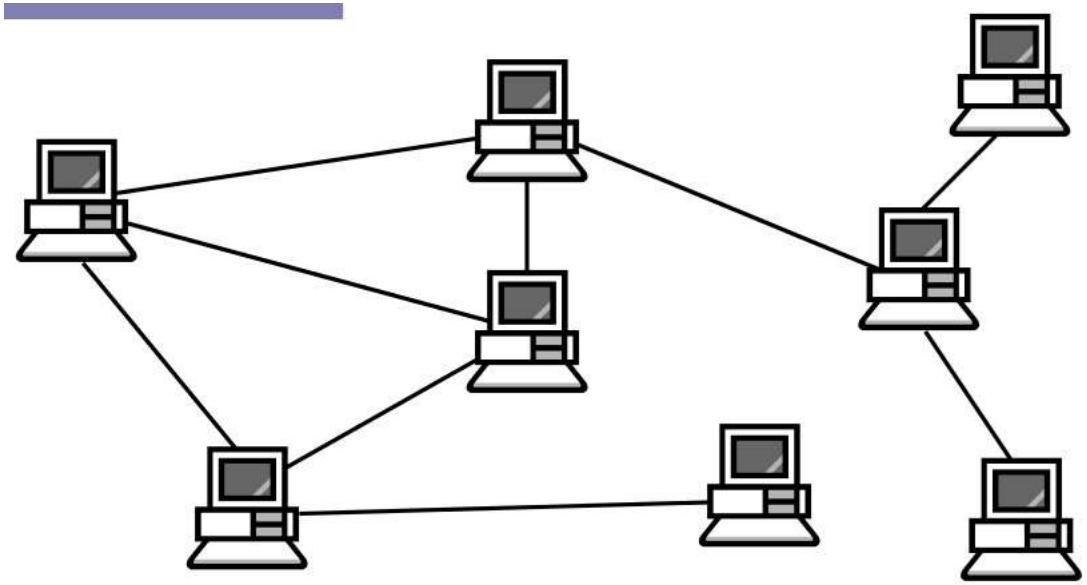
Jest szereg możliwych paradygmatów komunikacji w sieciach komputerowych. Jednym z najpopularniejszych jest architektura klient-serwer. Polega ona na zbudowaniu pewnej usługi sieciowej obsługiwanej przez jeden z węzłów sieci — serwer — świadczonej pod ustalonym adresem. W sieciach TCP/IP ten adres jest reprezentowany przez kombinację adresu IP i numeru portu, który jest lokalnym adresem usługi.



Komunikacja może odbywać się w trybie połączeniowym lub bezpołączeniowym. Klient przesyła serwerowi swoje zapytania lub polecenia, i związane z nimi dane, a serwer odsyła klientowi odpowiedzi lub potwierdzenia.

Modele komunikacji: peer-to-peer

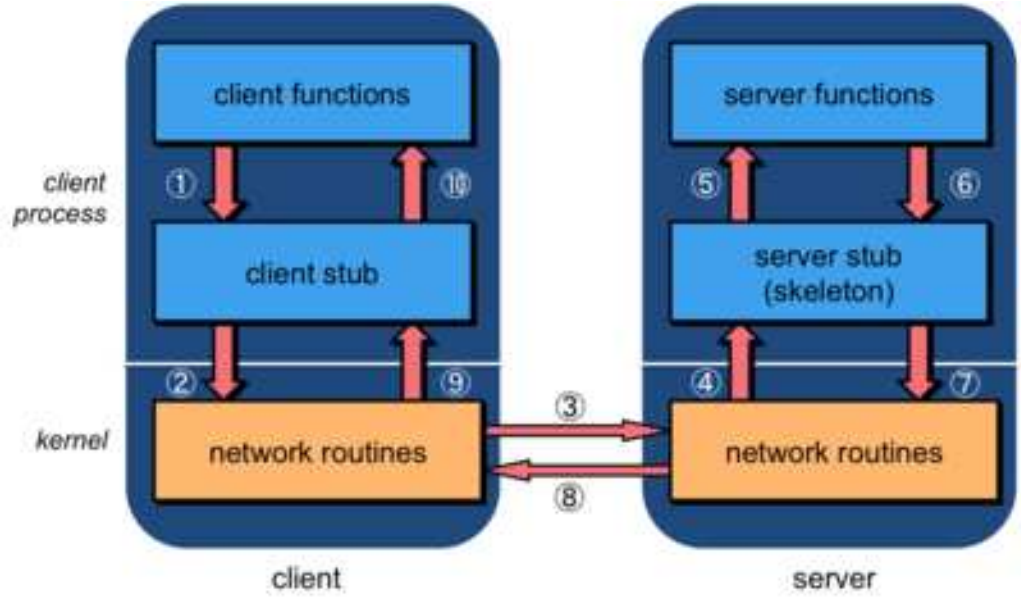
Alternatywną architekturą komunikacji w sieci jest struktura zwana *peer-to-peer*, w której węzły sieci komunikują się z wybranymi partnerami na zasadzie równorzędnej. Jednak ze względu na zarządzanie interfejsem sieciowym przez system operacyjny, na niskim poziomie rejestracja usługi/odbieranie komunikatów odbywa się w trybie serwera, a nawiązywanie połączenia/nadawanie komunikatów w trybie klienta.



Określenie *peer-to-peer* często stosuje się do protokołów komunikacyjnych wyższego rzędu, takich jak BitTorrent, IRC, Skype, itp. Jednak jest to również dobry model warstwy komunikacyjnej niskiego poziomu, w której może funkcjonować np. przemysłowy system sterowania.

Modele komunikacji: zdalne wywoływanie procedur

Innym paradygmatem komunikacji w sieciach komputerowych jest zdalne wywoływanie procedur (Remote Procedur Calling — RPC). W tym paradygmacie w ramach serwera instaluje się pewna liczba procedur, a program klienta je wywołuje. Technicznie program klienta wywołuje pewną szczątkową procedurę pośredniczącą — **namiastkę** (*stub*) należącą do biblioteki RPC i reprezentującą zdalną procedurę. Wywołanie namiastki powoduje przekazanie żądania wywołania procedury, wraz z jej argumentami, do zdalnego serwera, a tam analogiczna namiastka serwera wywołuje rzeczywistą procedurę w programie aplikacyjnym, a po jej zakończeniu odsyła obliczony wynik do węzła klienta. Technicznie wynik jest zwracany przez namiastkę.



Zdalne wywoływanie procedur — przekazywanie argumentów

Przekazywanie parametrów w wywołaniach RPC stwarza szczególne problemy. Wartości wielobajtowe mogą mieć inną reprezentację (*byte-order*) na maszynie wywołującej i obliczającej. Ponieważ ten problem występuje przy wszelkiego rodzaju komunikacji sieciowej, w stosie protokołów TCP/IP jest rozwiązany przez konwencję, że wielkości wielobajtowe są przesyłane przez sieć w porządku *big-endian*. Węzły sieci pracujące w architekturze *little-endian* muszą dokonać odwrócenia porządku przed i po transmisji.

To jednak nie rozwiązuje wszystkich problemów z przekazywaniem parametrów. Wartości złożone, takie jak struktury, mogą być inaczej upakowane, ponieważ program wywołujący na maszynie klienta mógł być skompilowany innym kompilatorem, niż program obliczający na maszynie serwera. Próba przekazania parametru lub wyniku funkcji przez adres lub referencję w ogóle nie ma sensu, ponieważ adres i referencja ma sens tylko w przestrzeni adresowej programu wywołującego.

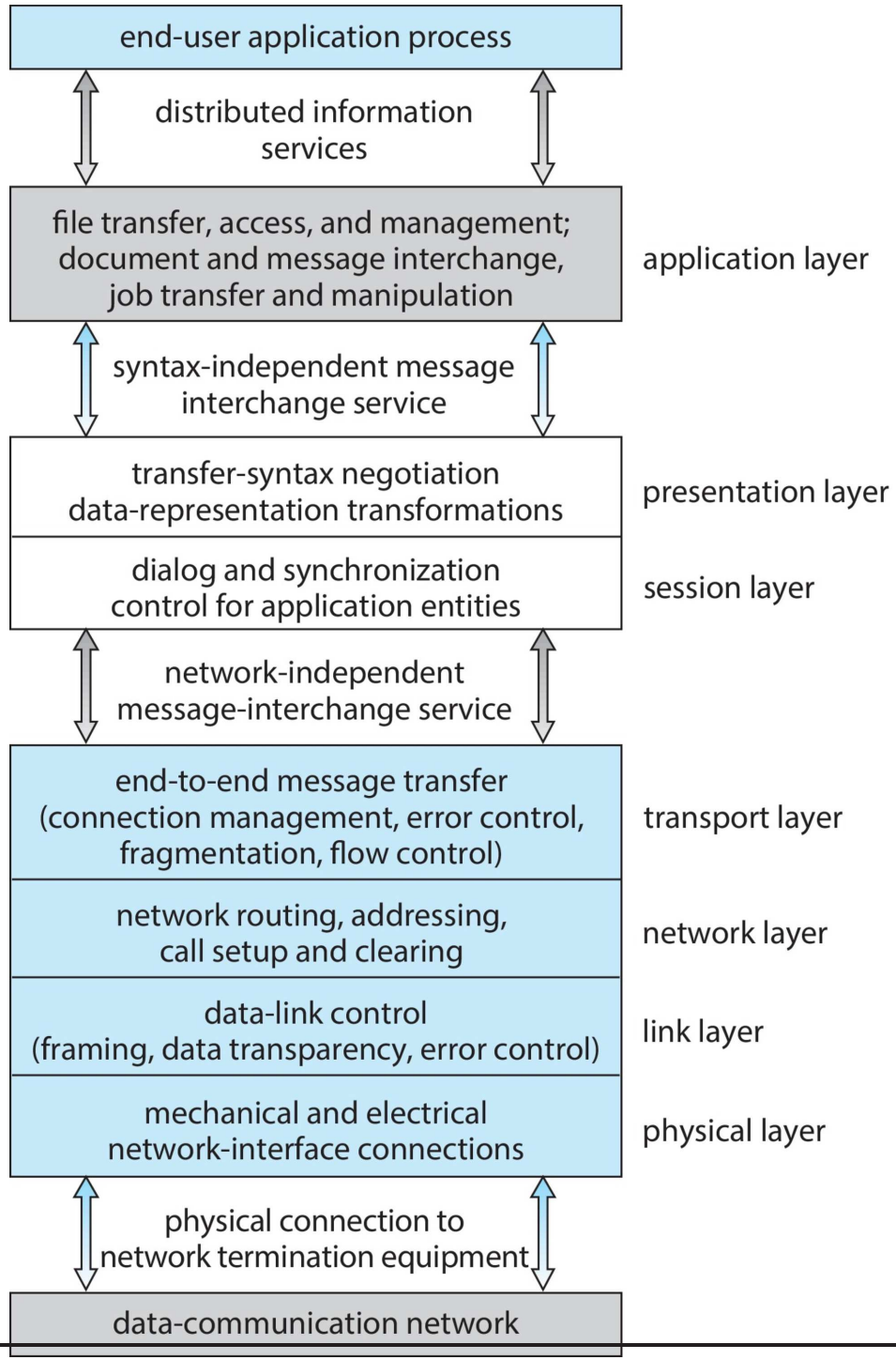
Dlatego w systemach RPC niezbędne jest specjalne przetwarzanie przekazywanych parametrów i wyników funkcji. Na przykład, system SunRPC zawiera specjalną specyfikację XDR (*eXternal Data Representation*), pozwalającą na **serializację** argumentów i wyników funkcji. Inne systemy pozwalają na jawne zakodowanie dowolnej struktury danych, np. ISO ASN.1 (*Abstract Syntax Notation*), JSON (*JavaScript Object Notation*), albo SOAP (*Simple Object Access Protocol*).

Sieci komputerowe w systemach operacyjnych

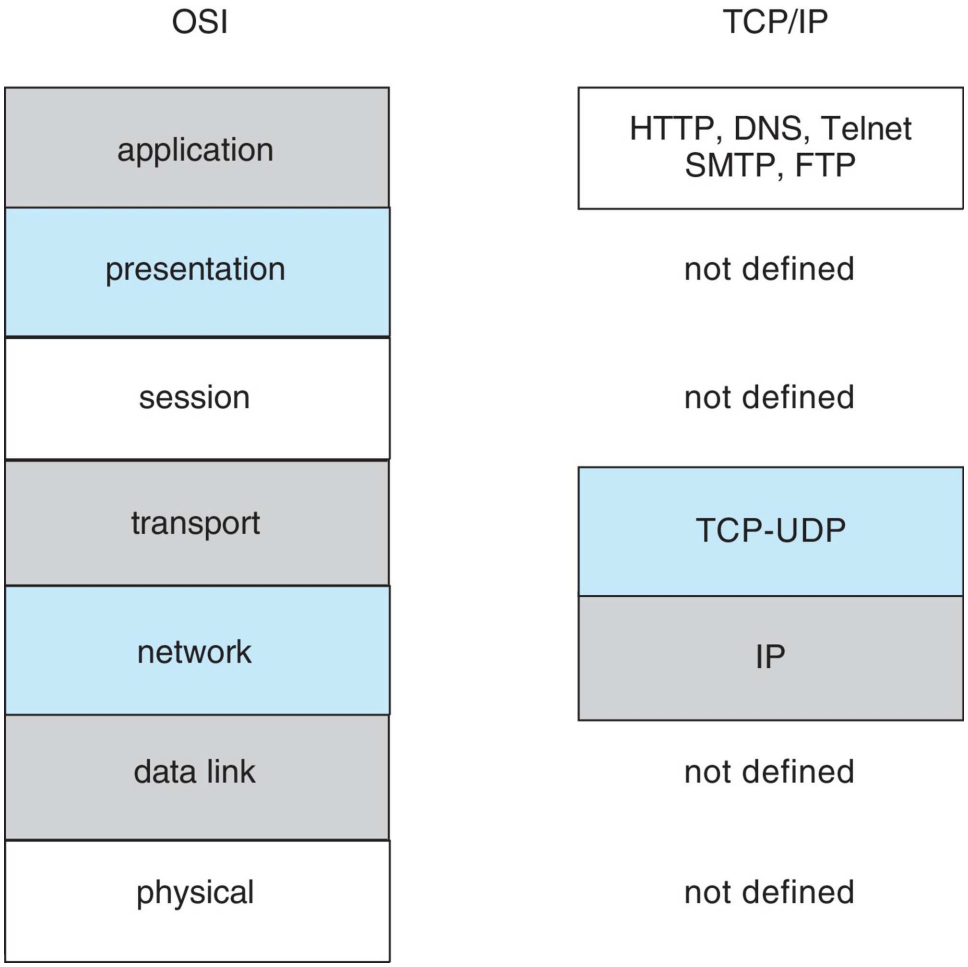
Systemy operacyjne realizują szereg funkcji w odniesieniu do sieci komputerowych:

1. Obsługa interfejsu sieciowego (karty sieciowej).
2. Implementacja protokołów komunikacyjnych (w jądrze systemu + biblioteka API).
3. Udostępnianie funkcji (API) do wysyłania pakietów sieciowych, inicjowania połączeń, i uzyskiwania dostępu do usług sieciowych.
4. Udostępnianie funkcji (API) do rejestrowania serwisów sieciowych obsługiwanych przez lokalny system.
5. Odbieranie pakietów otrzymanych przez interfejs sieciowy i ich obsługa:
   (a) przyjmowanie pakietów skierowanych do lokalnego systemu, przekazanie ich właściwemu odbiorcy (procedurze obsługowej),
   (b) obsługa pewnych niskopoziomowych protokołów sieciowych, takich jak: ARP, ICMP, itp.,
   (c) ewentualne funkcje zapory sieciowej (*firewall*).
6. Utrzymywanie tablicy ścieżek sieciowych i routing (wyznaczanie ścieżki sieciowej właściwej dla danego adresu).
7. Implementacja algorytmów translacji adresów symbolicznych DNS.

Warstwowy model sieci ISO/OSI



Warstwowy model sieci — odniesienie do protokołów internetowych



Routing (trasowanie?)

Routing jest czynnością określania dokąd należy wysłać pakiet sieciowy znając zawarty w nim adres odbiorcy. Tę czynność muszą wykonywać wszystkie komputery biorące udział w komunikacji sieciowej. Wymaga to lokalizacji sieci komputerowej na podstawie adresu. W praktyce wystarczy określić pierwszy węzeł na drodze do docelowego adresu.

Realizacja tej czynności opiera się na powiązaniu prefiksu adresu sieci z fizyczną lokalizacją sieci. Informacje wymieniane między komputerami obsługującymi ścieżki (routerami) pozwalają im na określanie tych ścieżek przez abstrakcję.

Algorytm routingu:

1. jeśli adres jest podsieci, do której jestem bezpośrednio włączony, to wysyłam pakiet do docelowego odbiorcy przez odpowiedni interfejs sieciowy;
2. jeśli znam ścieżkę do podsieci odbiorcy pakietu, określającą bramę lokalną (router) stanowiącą początek tej ścieżki, to przekazuję pakiet temu routerowi;
3. w przeciwnym przypadku wysyłka pakietu jest niemożliwe i należy go utylizować.

Szczególnym przypadkiem jest sytuacja, kiedy komputer posiada tzw. ścieżkę domyślną, określającą bramę dla wszystkich adresów, do których nie jest pamiętana indywidualna ścieżka. Jeśli komputer posiada zdefiniowaną taką ścieżkę to wie jak doręczyć wszystkie pakiety.

Routing — tablica ścieżek

Decyzja wyboru ścieżki sieciowej, do której należy wysłać dany pakiet sieciowy, jest podejmowana na podstawie docelowego adresu IP pakietu, i jej wynikiem jest wybór komputera w (jednej z) sieci lokalnej(ych), do której(ych) dany komputer jest podłączony. Routing jest czynnością wykonywaną w ramach protokołu IP (warstwy sieciowej, w nomenklaturze ISO/OSI).

Dla podejmowania tych decyzji system operacyjny utrzymuje **tablicę ścieżek** sieciowych, zawierającą wszystkie znane danemu systemowi ścieżki. Może istnieć wiele ścieżek w tej tablicy, i wybierana jest zawsze najlepiej dopasowana, tzn. najbardziej szczegółowa ścieżka zgodna z adresem docelowym. Komputery stanowiące zwykłe węzły klienckie często posiadają tylko ścieżkę do sieci lokalnej, oraz (opcjonalnie) ścieżkę domyślną. określającą gdzie należy wysyłać wszystkie pakiety adresowane poza sieć lokalną. Brak ścieżki domyślnej oznacza, że komputer może komunikować się tylko z partnerami w sieci lokalnej.

Dodatkowym parametrem każdej ścieżki jest jej **metryka** określająca koszt przesłania pakietu przez daną ścieżkę. Mogą istnieć różne ścieżki do tej samej sieci docelowej, ale z różnymi metrykami. W najprostszym przypadku metryka może określać liczbę segmentów sieci, które pakiet będzie musiał pokonać na drodze do sieci docelowej. Pokonanie każdego segmentu wiąże się z przetwarzaniem pakietu w jakimś urządzeniu, a więc im większa liczba segmentów tym dłużej będzie trwało przesyłanie pakietu.

Routing — tworzenie/obsługa ścieżek sieciowych

Skąd komputery biorą tablice ścieżek? Skąd komputer właśnie włączony do sieci może znać ścieżkę komputera z określonym adresem położonym np. w Australii? Zwłaszcza, że sieci komputerowe i połączenia między nimi zmieniają się dynamicznie, powstają nowe połączenia, znikają istniejące, zmieniają się metryki pewnych połączeń, występują awarie, itp.

Odpowiedzią na te pytania jest cały szereg dość złożonych procesów i technologii.

W niewielkich sieciach często stosuje się **routing statyczny** polegający na ręcznym tworzeniu i aktualizacji wszystkich ścieżek sieciowych danego systemu. W większych sieciach zwykle jest to niemożliwe, i stosuje się **routing dynamiczny**. Polega on na przekazywaniu informacji o zmianach ścieżek sieciowych przez komputery sobie nawzajem, i automatycznej aktualizacji tablicy ścieżek. Służą do tego specjalne protokoły komunikacji, pozwalające określić kto komu może przesyłać informacje o których ścieżkach, i od kogo można przyjmować wiążące informacje o zmianach.

Adresy symboliczne

Dla wygody wprowadzono dualną przestrzeń adresów — adresy symboliczne. Są one zorganizowane w hierarchiczną strukturę tzw. domen adresowych. Wszystkie domeny są własnością różnych organizacji, tworzących i wykorzystujących przestrzeń nazw w danej domenie. Struktura ta nie ma ograniczeń; w ramach każdej domeny można stworzyć kolejne domeny niższego poziomu, które same dostarczają informacji o adresach i domenach w nich zawartych.

Np.: komputer `sequoia.kcir.pwr.edu.pl` (IP:156.17.9.3) należy do domeny `kcir.pwr.edu.pl`, która jest własnością Katedry Cybernetyki i Robotyki Politechniki Wrocławskiej, i która otrzymała prawa do tej domeny od Politechniki Wrocławskiej, właściciela domeny `pwr.edu.pl`.

Przestrzeń adresów symbolicznych służy tylko wygodzie ludzkiej pamięci, i istnieje odwzorowanie adresów symbolicznych na adresy numeryczne. Do realizacji tego odwzorowania służy specjalny system DNS (*domain name system*) składający się z sieci serwerów wymieniających informacje o tych odwzorowaniach i serwujących te informacje na życzenie.

Translacja nazw symbolicznych — system DNS

- DNS (*Domain Name System*) — hierarchiczny, rozproszony system nazw symbolicznych w Internecie.
- Oparty na oddelegowaniu administracji domenami różnym instytucjom, korzystającym z własnych serwerów DNS, automatycznie wymieniającym między sobą informacje o administrowanych przez siebie domenach.

  domena — poddrzewo hierarchicznego drzewa nazw
- Własności: nadmiarowość, replikacja, buforowanie, duża niezawodność i tolerancja błędów, optymalizacja procesu uzyskiwania odpowiedzi w warunkach rzadkich zmian.
- Serwer DNS — program, którego zadaniem jest podawanie translacji adresu określonego w zapytaniu klienta, i komunikujący się z innymi serwerami DNS, w celu jej znalezienia.
- Serwery DNS mogą posiadać redundancję — dla danej domeny można wprowadzić oprócz głównego serwera **primary** równoważne serwery dodatkowe **secondary**. Ponadto istnieją serwery **caching-only**, który nie są oficjalnie zarejestrowanymi serwerami, ale odpowiadają na zapytania DNS, i przechowują wyznaczone translacje adresów, w celu przyspieszenia obsługi przyszłych zapytań.

Serwery systemu DNS

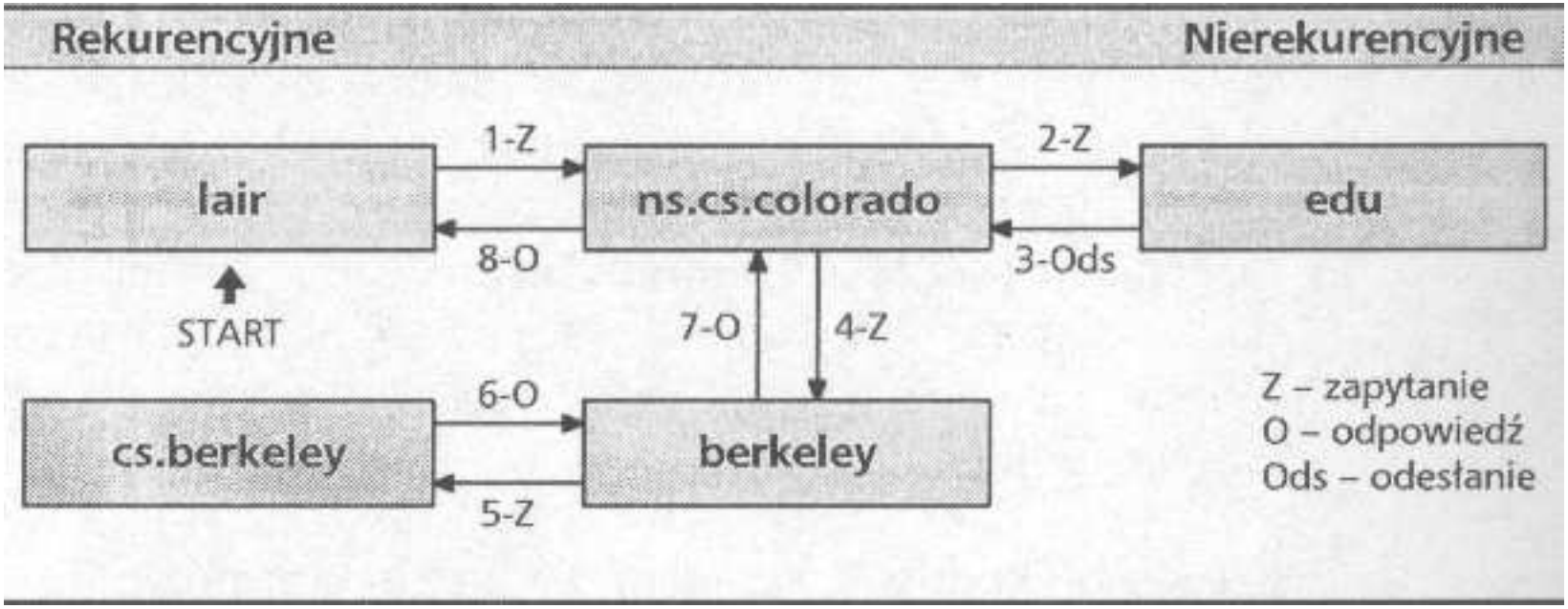
- serwer DNS domyślnie jest **rekurencyjny**; w sytuacji gdy nie zna odpowiedzi na otrzymane zapytanie, sam kontaktuje się z innymi serwerami aby ją uzyskać, i udzielić pytającemu klientowi

  rekurencyjny serwer jest właściwym rozwiązaniem dla sieci lokalnej, ponieważ pozwala klientom zawsze uzyskiwać odpowiedzi na swoje pytania, a ponadto może przechowywać uzyskane odpowiedzi, i udzielać ich potem kolejnym klientom bez ponownego odpytywania rekurencyjnego

- serwer DNS może być również **nierekurencyjny**; w przypadku nieznajomości odpowiedzi serwer taki nie pyta się innych serwerów, tylko odpowiada tzw. odsyłaczem (ang. *referral*), podającym adres innego, bardziej właściwego dla danej domeny serwera DNS

  serwery DNS wyższego poziomu w hierarchii Internetu (np. serwery główne takich domen jak .com albo .pl) są z zasady nierekurencyjne, więc tym bardziej nie przechowują informacji, które ich nie dotyczą

Przykład sekwencji odwołań do serwerów DNS dla zapytania o nazwę `mammoth.cs.berkeley.edu` wykonanego na komputerze `lair.cs.colorado.edu`:



Dla domen pośrednich pomiędzy siecią lokalną a domeną główną Internetu musimy wybrać pomiędzy pracą rekurencyjną a nierekurencyjną serwera DNS. Jednak nierekurencyjny serwer nie może obsługiwać normalnych klientów, nieprzygotowanych na otrzymanie na swoje zapytanie odpowiedzi w postaci odsyłacza.

Systemy rozproszone

Systemem rozproszonym nazywamy zbiór węzłów obliczeniowych dowolnie połączonych siecią komunikacyjną. Węzły te mogą różnić się swoimi możliwościami, zasobami, a także architekturą sprzętową oraz oprogramowaniem, w tym również systemami operacyjnymi. Mogą posiadać różne procesory różniące się zestawem instrukcji, długością słowa, a nawet organizacją bajtów w słowie (tzw. *byte-order*).

Organizacja „big-endian" oznacza, że bajty w słowie uporządkowane są od najstarszego do najmłodszego (np.: Motorola 68000, SPARC), natomiast „little-endian" oznacza bajty uporządkowane od najmłodszego do najstarszego (np.: Intel 80x86).[1] Ponieważ komunikacja sieciowa odbywa się w układzie strumienia bajtów, pomiędzy systemami o różnej organizacji nie można bezpośrednio przesyłać wartości wielobajtowych.

[1] Nowsze generacje procesorów posiadają rozkazy umożliwiające zarówno pobrania jak i zapisy w wybranej kolejności bajtów, np.: MIPS, PowerPC, SPARCv9, ARM od v3.

Zastosowania systemów rozproszonych

współdzielenie zasobów — systemy plików, bazy danych, specjalistyczne procesory (superkomputery wektorowe, procesory graficzne), drukarki, itp.

przyspieszenie/zrównoleglenie obliczeń/równoważenie obciążeń

zwiększenie niezawodności — gdy zasoby albo procesy są zdublowane
uwaga: gdy zasoby są zróżnicowane i pełnią różne funkcje, niezawodność systemu rozproszonego nie wzrasta

Sieciowe i rozproszone systemy operacyjne

Pod względem obsługi sieci komputerowych rozróżnia się dwa rodzaje systemów operacyjnych:

1. sieciowe systemy operacyjne,

2. rozproszone systemy operacyjne.

Sieciowe systemy operacyjne zapewniają obsługę sieci komputerowej, oferując użytkownikom system dostęp do mechanizmów sieciowych. **Rozproszone systemy operacyjne** wykorzystują mechanizmy sieciowe aby zaprezentować użytkownikowi spójny widok zasobów dostępnych przez sieć.

Sieciowe systemy operacyjne

Sieciowy system operacyjny tworzy środowisko pozwalające użytkownikom na dostęp do zasobów sieciowych, np. przez logowanie się do zdalnych serwerów, transferowanie plików, lub korzystanie z innych serwisów sieciowych. Wszystkie współczesne systemy operacyjne ogólnego przeznaczenia należą do tej grupy, np. Linux, Windows, Android, itp.

Korzystanie z usług sieciowych wymaga od użytkowników pracy w środowisku danej usługi, np. posługiwania się poleceniami `FTP`, lub poleceniami zdalnego systemu operacyjnego w czasie zdalnej sesji `ssh`. Niektóre z usług sieciowych mogą zapewniać interfejs graficzny. Niektóre mogą również oferować programistyczne API pozwalające na pisanie programów korzystających z danej usługi sieciowej.

Rozproszone systemy operacyjne

W rozproszonym systemie operacyjnym użytkownik korzysta ze zdalnych zasobów w taki sam sposób jak z lokalnych. System może implementować:

migrację danych — gdy dane przechowywane w węźle A potrzebne są w węźle B, muszą zostać przesłane do B w postaci całego pliku, lub jego fragmentu; po zakończeniu przetwarzania muszą one zostać przesłane z powrotem do A, o ile zostały zmodyfikowane

migrację obliczeń — gdy pewne obliczenia dotyczą większej ilości danych, może bardziej się opłacać wykonać je na węźle, gdzie dane są przechowywane; można to zrealizować za pomocą zdalnego wywołania procedur RPC, albo można stworzyć protokół zdalnych obliczeń, w którym proces wysyła komunikat do zdalnego systemu, który tworzy odpowiednie procesy obliczeniowe, i odsyła wyniki gdy one się zakończą

migrację procesów — rozszerzenie migracji obliczeń do całego procesu; implementacja może być przezroczysta i niewidoczna dla inicjującego procesu (metoda zwykle stosowana w celu wyrównania obciążeń i/lub przyspieszenia obliczeń w kompatybilnych systemach), albo może być specjalnie zaprogramowana przez proces inicjujący (metoda zwykle stosowana w celu wykorzystania zdalnego specjalnego sprzętu lub oprogramowania)

Niezawodność w systemach rozproszonych

Systemy rozproszone są szczególnie czułe na różnego rodzaju awarie. Z definicji wykorzystują one duży zestaw sprzętu, rozmieszczonego i administrowanego w oddalonych lokalizacjach, oraz sieci komunikacyjnych cechujących się nieprzewidywalnością czasu transmisji i opóźnień.

Ponieważ użytkownik nie ma wiedzy ani świadomości o procesach realizujących rozproszone operacje, wszelkie awarie nawet peryferyjnych elementów systemu, albo opóźnienia komunikacyjne, będzie on odczuwał jako niepoprawne działanie systemu. Dlatego niezawodność jest istotnym czynnikiem funkcjonowania systemów rozproszonych.

Terminologia niezawodności

Defektem (*fault*) nazywamy wadę wadę systemu, taką jak błędny fragment kodu, lub nieprawidłowo działające urządzenie. Istnienie defektu nie oznacza jeszcze, że błędny kod zostanie kiedykolwiek wykonany, albo urządzenie zgubi lub zniekształci jakieś dane.

Błędem (*error*) nazywamy sytuację, gdy jakiś element systemu — program lub urządzenie — działa niepoprawnie, np. program odwołuje się do adresu spoza dozwolonego zakresu, albo urządzenie przekłamuje niektóre dane w transmisji. Błędy takie mogą być jednak zauważone przez inne elementy systemu, które mogą spowodować restart programu lub retransmisję pakietu danych.

Awarią (*failure*) nazywamy sytuację, kiedy system nie jest w stanie realizować swojej/swoich funkcji wskutek wystąpienia błędu.

Odporność na defekty (*fault tolerance*) to zdolność systemu do poprawnej pracy w sytuacji gdy istnieje w nim defekt, a także gdy powstał błąd.

Wykrywanie awarii

W przypadku systemów rozproszonych defekty mogą powstawać zarówno w łączach komunikacyjnych tworzących strukturę sieci, jak i w jej konkretnych węzłach.

Wykrywanie awarii jest możliwe przez komunikację pomiędzy węzłami. Węzły mogą wysyłać sobie komunikaty kontrolne typu: „Pracuję poprawnie.” albo „Czy pracujesz poprawnie?”

W ogólnym przypadku może jednak nie być możliwe odróżnienie następujących sytuacji:

- zdalny węzeł uległ awarii,
- łącze komunikacyjne do zdalnego węzła uległo awarii,
- komunikat kontrolny został zgubiony.
- zdalny węzeł oraz łącze komunikacyjne są sprawne, ale komunikat kontrolny został przetrzymany i opóźniony ponad dopuszczalny limit.

Rekonfiguracja i odbudowa po awarii

Redundancja (nadmiarowość) jest jedną z głównych technik budowania odporności na defekty, zarówno w sprzęcie jak i oprogramowaniu. W oczywisty sposób, ponieważ prowadzi ona do budowy bardziej złożonych systemów, może sama w sobie wprowadzać ryzyko dalszych defektów i związanych z nimi awarii.

W momencie wykrycia awarii zdalnego węzła (praktycznie: braku komunikacji ze zdalnym węzłem), jeśli system posiada redundancję, powinien dokonać **rekonfiguracji** aby kontynuować normalną pracę. Może to polegać na wyborze alternatywnej ścieżki komunikacji sieciowej, lub zaprzestaniu korzystania z usług zdalnego systemu i zastąpienie ich usługami świadczonymi przez inny element systemu.

Po ustaniu lub likwidacji awarii system rozproszony powinien wrócić do pełnej konfiguracji. Przywrócenie sprawności łącza komunikacyjnego albo zdalnego systemu można wykryć za pomocą komunikatów kontrolnych. Odbudowany/zrestartowany węzeł sieci musi oczywiście zaktualizować swoją wiedzę o stanie całego systemu i wykonywanych w nim procesów.

Skalowalność

Skalowalność jest zdolnością systemu do dostosowywania się do zwiększonego obciążenia. Skalowalność jest pojęciem względnym, ponieważ wszystkie systemy obliczeniowe mają ograniczoną pojemność, szybkość, przepustowość, i w warunkach zwiększonego obciążenia mogą ulec całkowitemu nasyceniu. Jednak system skalowalny reaguje w bardziej akceptowalny sposób, jego wydajność zmniejsza się stopniowo, i później osiąga stan całkowitego nasycenia niż system nieskalowalny.

Systemy rozproszone mają zdolność wykorzystania większych zasobów, i zwiększenia wydajności obliczeń, zatem powinny one być skalowalne w większym stopniu niż indywidualne komputery. Skalowalność jest złożonym zagadnieniem. Można ją budować za pomocą zapasowych zasobów (komputerów, łączy komunikacyjnych), jednak sama redundancja nie gwarantuje skalowalności. Np. gdy jeden węzeł zostanie nasycony procesami obliczeniowymi, i przestanie odpowiadać, uruchomienie wszystkich tych obliczeń na węźle zapasowym spowoduje i jego nasycenie i nic nie rozwiąże.

Skalowalność jest związana z odpornością na defekty. Redundancja istniejąca w systemie rozproszonym przyczynia się zarówno do jego większej niezawodności, jak i umożliwia rozładowanie szczytowych obciążeń. Kluczowe jest jednak użycie właściwych technologii.

Rozproszone systemy plików

W kontekście systemu plików, **usługa** (*service*) jest systemem oprogramowania, wykonującym się na jednym lub wielu węzłach sieci komputerowej, realizującym pewną funkcjonalność dla klientów. **Serwer** jest programem wykonywanym na jednym węźle sieci, prezentującym interfejs do funkcji systemu plików. **Klient** jest procesem, który może wywoływać funkcje systemu plików korzystając z tego interfejsu.

Rozproszony system plików (*distributed file system*, DFS) jest systemem, którego elementy: klienci, serwery, oraz urządzenia pamięci masowej (dyski magnetyczne, SSD, itp.) znajdują się na wielu (więcej niż jednym) węzłach sieci komputerowej. Te węzły mogą wykonywać różne wersje oprogramowania, nawet pod kontrolą różnych systemów operacyjnych. Jednocześnie DFS prezentuje użytkownikowi (lub aplikacji na węźle klienckim) widok systemu plików taki, jakby to był system plików na lokalnym dysku.

Głównym parametrem wydajności systemów DFS jest czas obsługi żądania klienta. W systemie lokalnym składa się on głównie z czasu dostępu do dysku, plus typowo znacznie krótszego czasu przetwarzania żądania. W systemie rozproszonym dochodzą do tego narzuty związane z komunikacją sieciową: czas przesłania żądania na zdalny serwer, czas przetwarzania przez warstwy protokołu komunikacyjnego, oraz czas przesłania odpowiedzi, uwzględniający przesłanie wszystkich danych pliku,

Implementacja systemów DFS

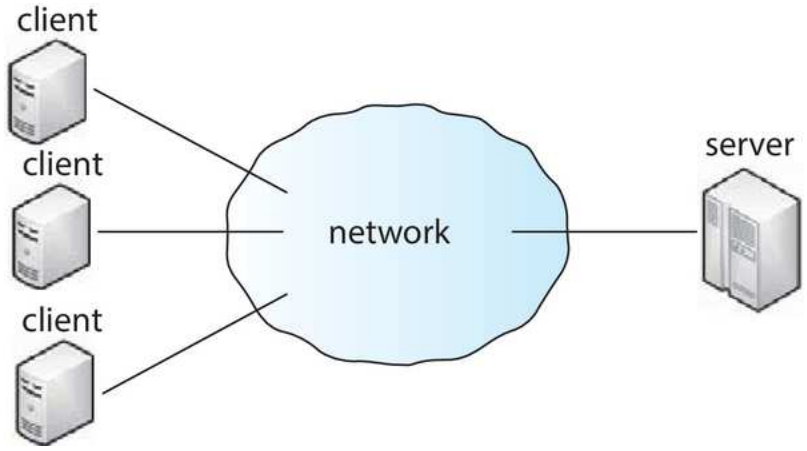
Można wyróżnić dwa modele implementacji systemów DFS: **model klient-serwer**, i **model oparty na klastrach**.

Model klient-serwer odpowiada sytuacji gdzie pewna liczba klientów korzysta z jednego lub więcej systemów plików, i pliki są udostępniane klientom, po jednym na raz. Przykładami takich systemów DFS są: Network File System (NFS) i Andrews File System (OpenAFS).

Model oparty na klastrach jest odpowiedni dla sytuacji gdzie wielu klientów korzysta z wielkich zbiorów danych i wymagana jest duża wydajność i skalowalność. Przykładami takich systemów są: Google file system i otwarty system HDFS (część systemu Hadoop).

Model klient-serwer DFS

W modelu klient-serwer pliki lub systemy plików są przechowywane na systemie pamięci masowej serwera. Typowo, stanowią one lokalny system plików w ramach serwera, natomiast system DFS istnieje jako usługa sieciowa.



Klienci łączą się z serwerem przez sieć za pomocą dedykowanego protokołu. Serwer weryfikuje uprawnienia klienta i realizuje żądania albo dostarczając plik klientowi, albo przyjmując zmiany treści lub metadanych pliku. Istotnym zagadnieniem jest utrzymanie spójności kopii pliku przechowywanych przez serwer i klienta, zwłaszcza w kontekście możliwości jednoczesnego dostępu do pliku przez wielu klientów.

W oczywisty sposób, praca systemu DFS modelu klient-serwer zostaje zatrzymana w przypadku awarii serwera.

Model klient-serwer DFS — przykłady

Network File System (NFS) opracowany przez Sun Microsystems jako protokół otwarty w latach 1980-tych był uniwersalnie wdrożony w całym świecie systemów uniksowych. Głównym celem tego protokołu była niezawodność po awarii serwera. Protokół NFS został zaprojektowany jako bezstanowy. To znaczy, serwer nie przechowuje informacji o tym, którzy klienci mają otwarte które pliki. Każde żądanie klienta zawiera pełną informację jaka operacja ma być wykonana na którym pliku. Dla obsługi jednoczesnej pracy wielu klientów na jednym pliku zaimplementowany został oddzielny protokół blokowania plików.

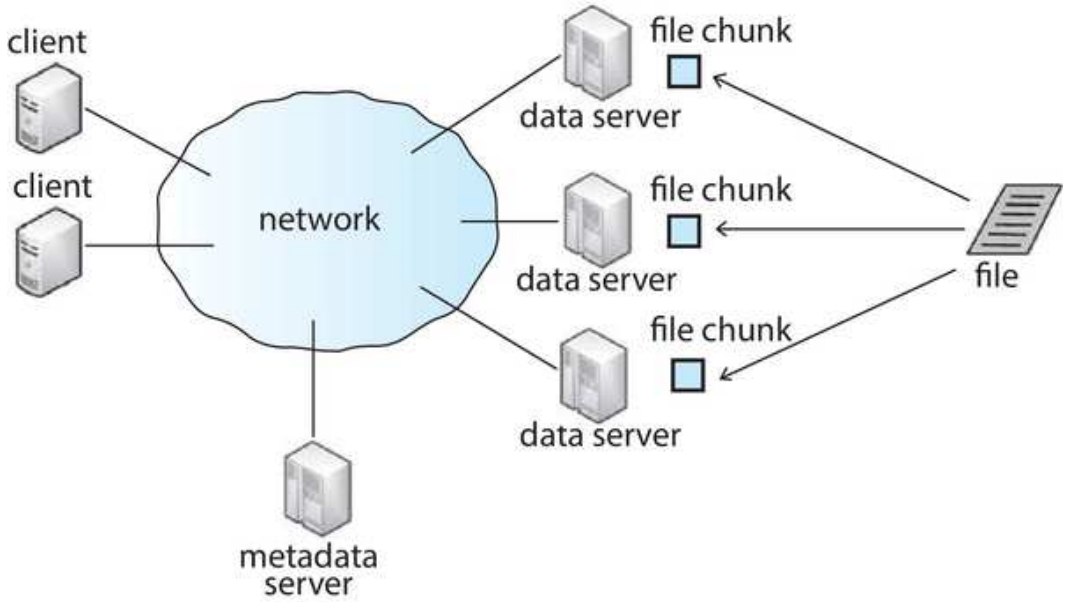
NFS nie był zaprojektowany dla wysokiej wydajności w szybkich sieciach. Jego nowsze wersje poprawiają tę wydajność przez szereg zmian technologicznych.

Andrew File System (OpenAFS) powstał z myślą o wydajności i skalowalności na wielką liczbę klientów. Minimalizuje ruch sieciowy pomiędzy serwerem a klientami. Plik jest przesyłany do klienta w momencie jego otwarcia, a zmiany odsyłane na serwer w chwili zamknięcia pliku przez klienta.

Innym przykładem systemu DFS modelu klient serwer jest system SMB (*Server Message Block*) pierwotnie opracowany przez IBM, wykorzystywany w systemach Windows, i rozwinięty do otwartej (niewłasnościowej) specyfikacji CIFS (*Common Internet File System*). Choć pierwotnie zaprojektowany dla sieci LAN, kolejne wersje cechują się lepszą wydajnością w sieciach WAN.

Model DFS oparty na klastrach

W nowoczesnych systemach szybkiego przetwarzania wielkiej ilości danych powstaje zapotrzebowanie na wydajny i skalowalny system DFS odporny na awarie sieci i serwerów. Dla takich zastosowań powstał model DFS oparty na klastrach.



Google File System (GFS) i Hadoop Distributed File System (HDFS) działają zgodnie z przedstawionym schematem. W tych systemach klienci łączą się z **serwerem metadanych**, który nie przechowuje rzeczywistych danych, a jedynie strukturę systemu plików (katalogi), oraz informacje o tym na których z wielu serwerów danych znajdują się które (kilkakrotnie replikowane) fragmenty (*chunk*) poszczególnych plików.

Model DFS oparty na klastrach — uwagi

Po pierwszym zapytaniu do serwera metadanych, klient uzyskuje informacje o lokalizacji serwerów danych zawierających poszczególne fragmenty.

Typowe systemy DFS oparte na klastrach są optymalizowane na operacje odczytu, i dopisywania na końcu pliku (*append*). **Nie umożliwiają one operacji bezpośredniego zapisu** (*random write*). Powoduje to zmniejszenie obciążenia serwera metadanych, z którym zwykle klienci mogą kontaktować się tylko raz, po czym dalsze interakcje realizują z serwerami danych.

Serwer metadanych jest natomiast odpowiedzialny za rozproszenie fragmentów pomiędzy serwery danych.

Systemy DFS oparte na klastrach nie prezentują również interfejsu systemu plików zgodnego z POSIX, a więc nie pozwalają one na odwzorowanie katalogów na maszynach klienckich. Zamiast tego systemy klientów muszą instalować dedykowane oprogramowanie umożliwiające dostęp to plików w takich DFS.