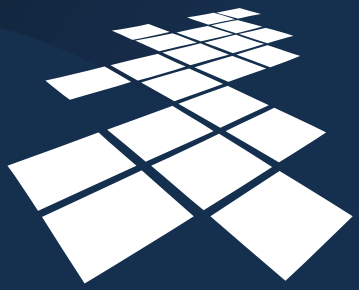
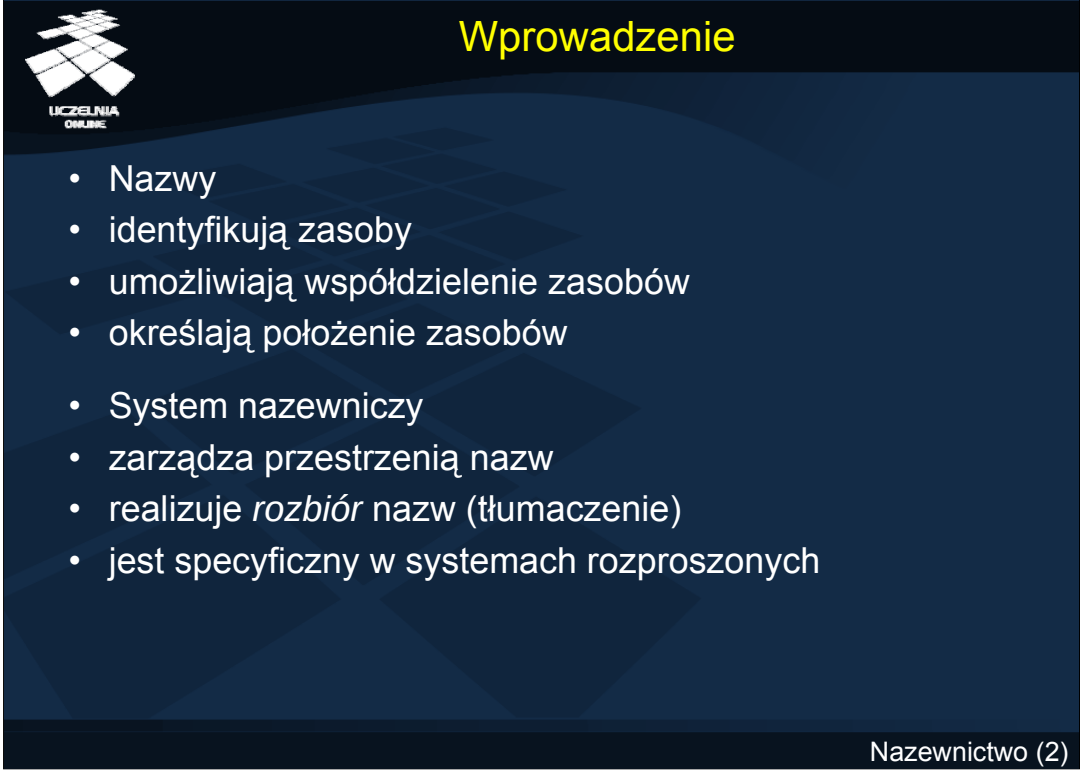
Systemy rozproszone

Nazewnictwo

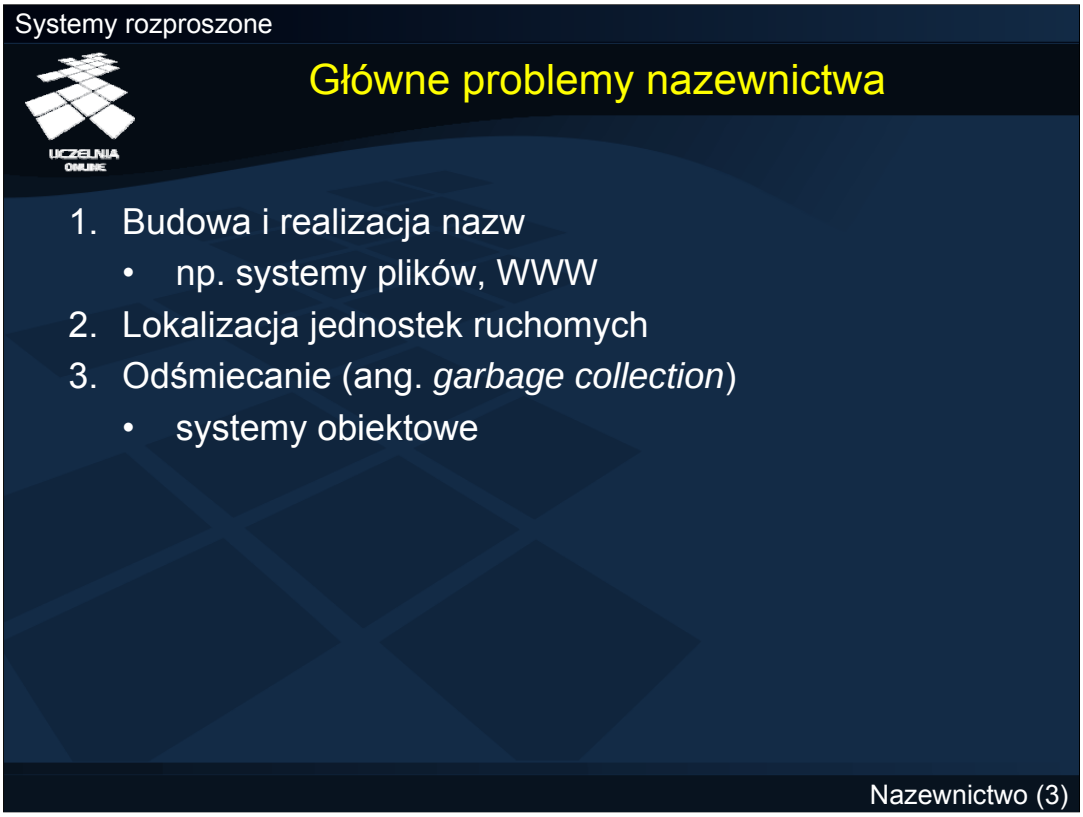
Cezary Sobaniec



UCZELNIA
ONLINE



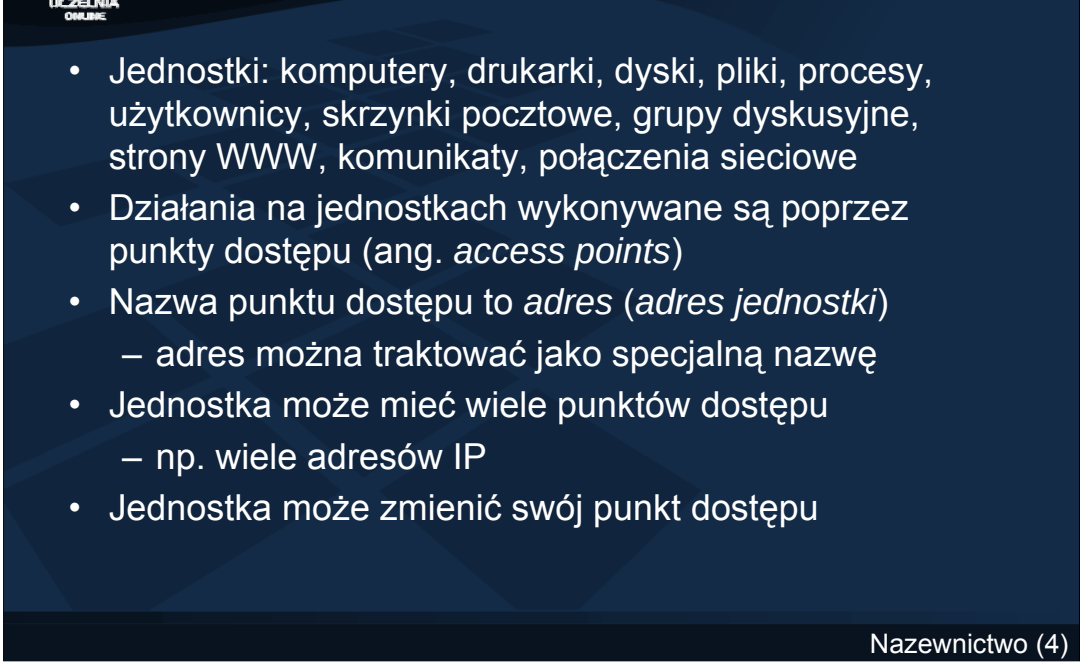
Zasoby muszą być w systemie jednoznacznie identyfikowane. Identyfikatory niskiego poziomu (systemowe) są trudne do zapamiętania, niewygodne i zależne od aktualnie stosowanej technologii. Stąd potrzeba wprowadzenia *nazw*, które będą wygodniejsze w użyciu, i które ukryją przed użytkownikiem szczegóły organizacji wewnętrznej systemu rozproszonego. Odwołanie dwóch rozproszonych procesów do zasobu o tej samej nazwie umożliwia jego dzielenie. Rozbiór (tłumaczenie) nazwy zasobu pozwala na jego zlokalizowanie. Nazewnictwo – jako zadanie systemu operacyjnego – występują również w systemach scentralizowanych, ale systemy rozproszone wymagają specjalizowanej realizacji systemu nazewniczego. Realizacja ta jest sama w sobie rozproszona, co jest konieczne dla osiągnięcia wysokiej efektywności i skalowalności tej usługi.



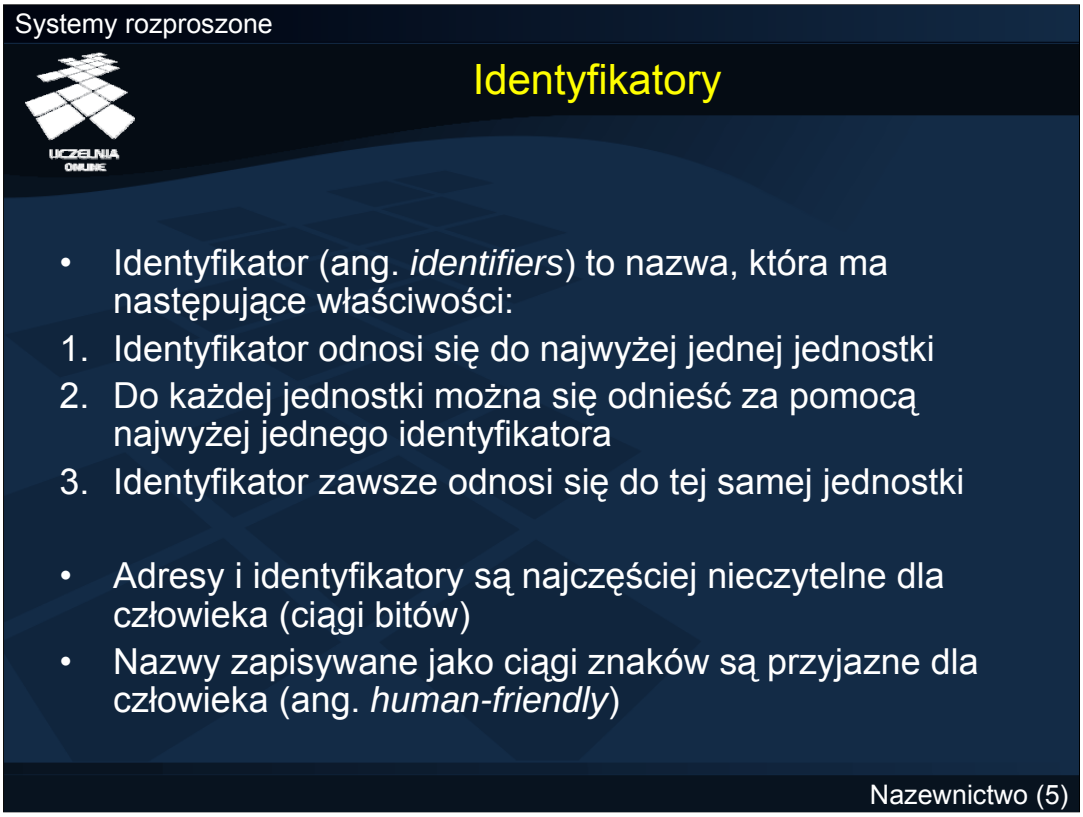
Nazwy powinny być przyjazne dla użytkownika, stąd ich konstrukcja staje się ważnym zagadnieniem projektowym, tym bardziej, że błędne założenia mogą np. ograniczyć skalowalność systemu.

System nazewnictwa musi być w stanie obsłużyć bardzo dużą (i ciągle wzrastającą) liczbę jednostek. Problem skalowalności może szczególnie dać znać o sobie w systemach, gdzie lokalizacja jednostki może ulegać zmianie. W konsekwencji nie można np. bezproblemowo powielać informacji o aktualnym adresie, gdyż może on się stać nieaktualny w krótkim czasie.

Trzeci istotny problem nazewnictwa dotyczy zarządzania przestrzenią nazw, a szczególnie nazwami, do których nie ma już odniesień. Nazwy takie powinny być wykrywane, a zasoby związane z tymi nazwami zwalniane. Zagadnienie to jest szczególnie istotne w systemach obiektowych.



Stosowanie nazw jest wygodne ponieważ uniezależnia nas od zmieniających się adresów. Przykładem może być posługiwanie się nazwą serwera FTP zamiast jego adresem IP. Zmiana lokalizacji serwera i rekonfiguracja systemu nazewniczego powodują, że serwer jest nadal dostępny pod tą samą nazwą, pomimo zmiany adresu IP (czyli punktu dostępu). Nazwy, które nie zawierają w sobie elementów adresów są określane jako nazwy **niezależne od położenia** (ang. *location independent*).

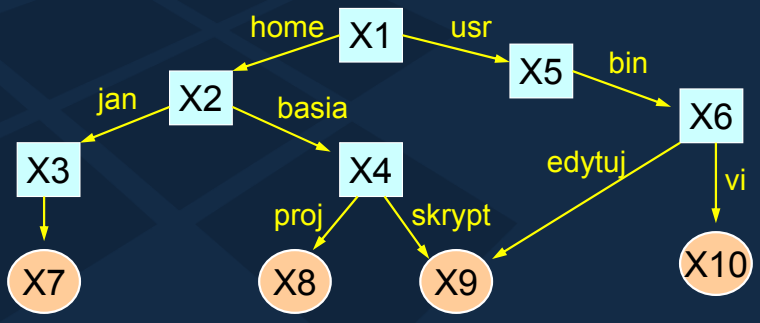


Identyfikatory jednoznacznie identyfikują jednostki i nie zmieniają się. Zwykłe nazwy nie są tak jednoznaczne. Aby upewnić się, że dwa odwołania dotyczą tego samego zasobu wystarczy porównać identyfikatory występujące w tych odwołaniach. Adresy również, pomimo, że są unikalne, nie spełniają warunków wymaganych dla identyfikatorów, ponieważ mogą być powtórnie użyte dla oznaczenia innej jednostki (np. adres IP może być przypisany innemu komputerowi).

Adresy i identyfikatory są najczęściej reprezentowane w postaci łańcuchów bitów, a więc postaci nieczytelnej dla człowieka. Przykładem może być adres IP (32 bity), adres w sieci Ethernet (48 bitów), czy adres komórki w pamięci (np. 64 bity).

Przestrzenie nazw

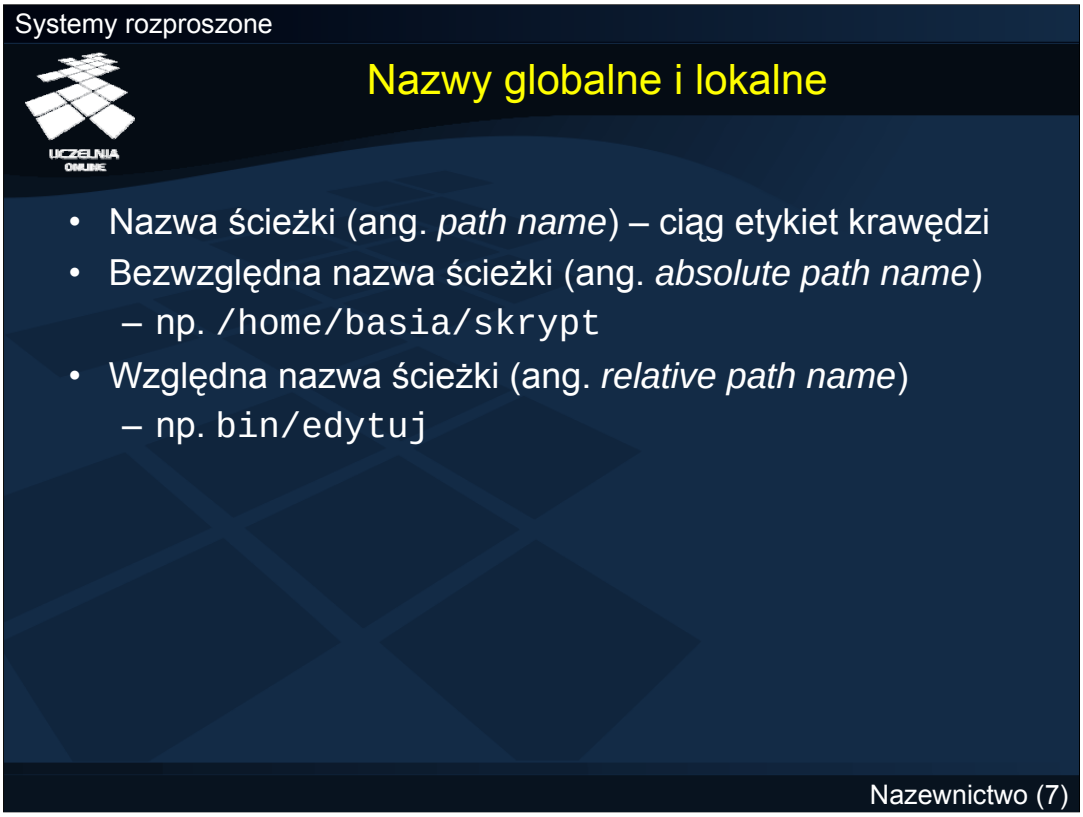
- Przestrzeń nazw (ang. *name space*) to uporządkowany zbiór wszystkich nazw w ramach usługi
- Przestrzeń nazw jest zorganizowana w postaci grafu skierowanego



Przestrzeń nazw można zobrazować za pomocą zaetykietowanego grafu skierowanego z dwoma rodzajami węzłów. Węzły-liście (ang. *leaf nodes*) reprezentują poszczególne jednostki i nie wychodzą z nich żadne nowe krawędzie. Węzeł taki przechowuje informacje o danej jednostce: jej adres, stan. Na rysunku węzły-liście zostały zaznaczone jako kółka. Drugim rodzajem węzłów są węzły katalogowe (ang. *directory node*), które mają zaetykietowane krawędzie przychodzące i wychodzące. Każdy węzeł katalogowy jest oddzielną jednostką w systemie rozproszonym i posiada swój identyfikator. Krawędzie wychodzące są zaetykietowane i wskazują na inne jednostki poprzez ich identyfikatory. Na rysunku węzły katalogowe zostały zaznaczone w kwadratach.

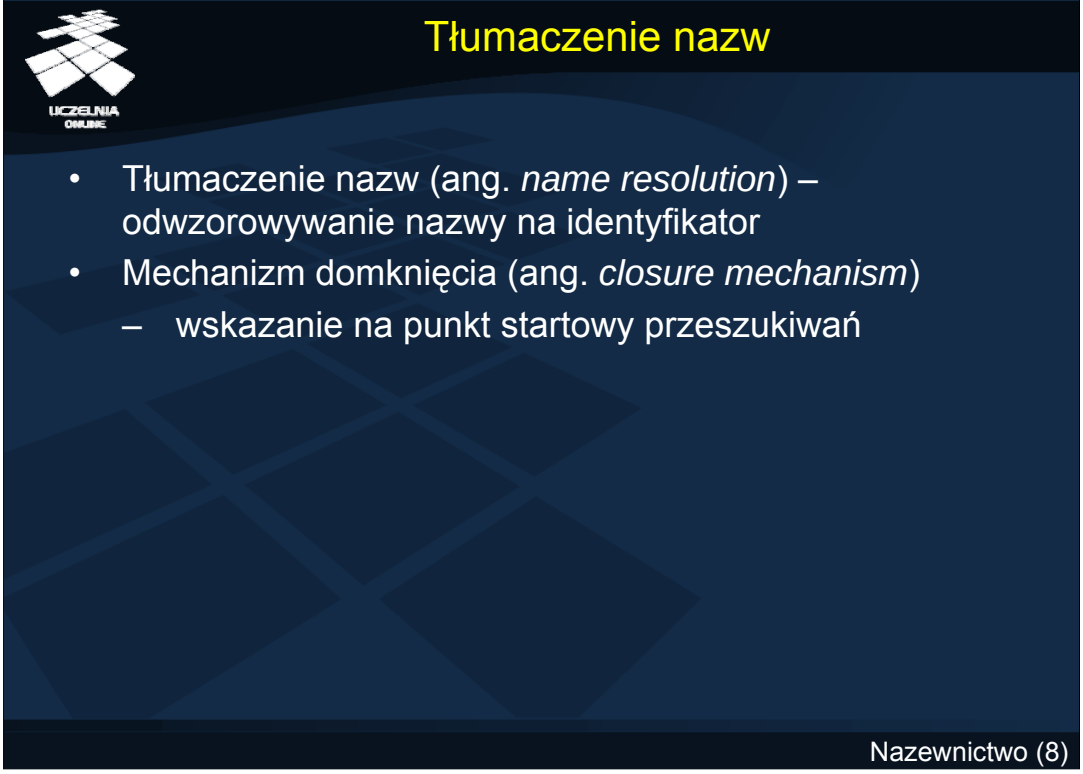
Jeden z węzłów w przedstawionym grafie nie posiada krawędzi przychodzących (węzeł X1). Jest to tzw. korzeń (ang. *root node*) grafu nazewniczego. Z reguły w systemach nazewniczych występuje tylko jeden korzeń.

Przedstawiony rysunek jest przykładem organizacji nazw dla systemu plików. Węzły-liście to pliki a węzły katalogowe to po prostu katalogi.

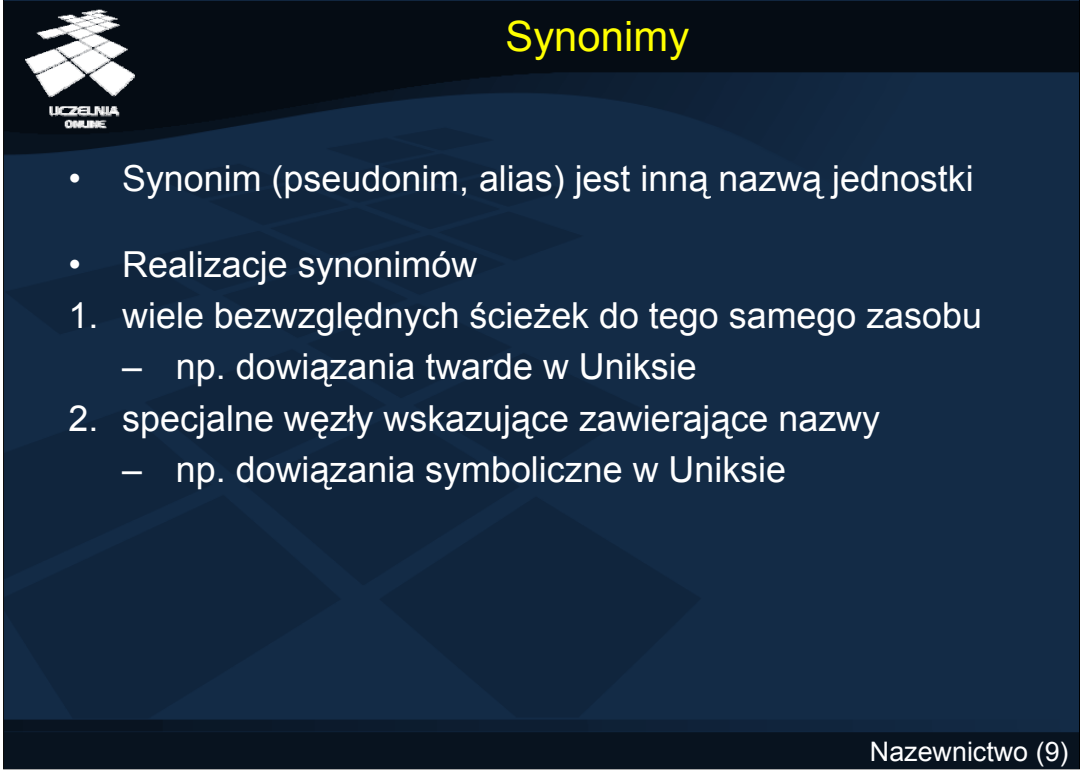


Każda ścieżka w grafie może być zapisana jako ciąg etykiet odpowiednich krawędzi wraz z etykietą węzła startowego. Taki ciąg nazywamy *nazwą ścieżki*. Jeżeli nazwa ścieżki rozpoczyna się od korzenia, to jest to **ścieżka bezwzględna** (globalna). Nazwy ścieżek nie rozpoczynające się od korzenia będą określane jako **nazwy względne** (względem węzła startowego) lub nazwy lokalne.

W systemach plików do zapisu nazw plików stosuje się notację korzystającą z ukośników do oddzielania nazw poszczególnych krawędzi w grafie. Zapis */home/basia/skrypt* oznacza więc odwołanie do pliku *skrypt* znajdującego się w katalogu *basia*, który z kolei jest podkatalogiem katalogu *home*, a ten jest bezpośrednim podkatalogiem katalogu głównego (korzenia systemu plików). Zapis „*/*” oznacza tu korzeń. Warto zauważyć, że ten sam plik jest dostępny również pod inną nazwą bezwzględną: */usr/bin/edytuj*. W tym przypadku graf jest grafem skierowanym acyklicznym. Niektóre usługi nazewnicze dopuszczają jednie ściśle hierarchiczną postać, w więc graf nazw jest tam drzewem (każda jednostka ma dokładnie jedną bezwzględną nazwę ścieżki).

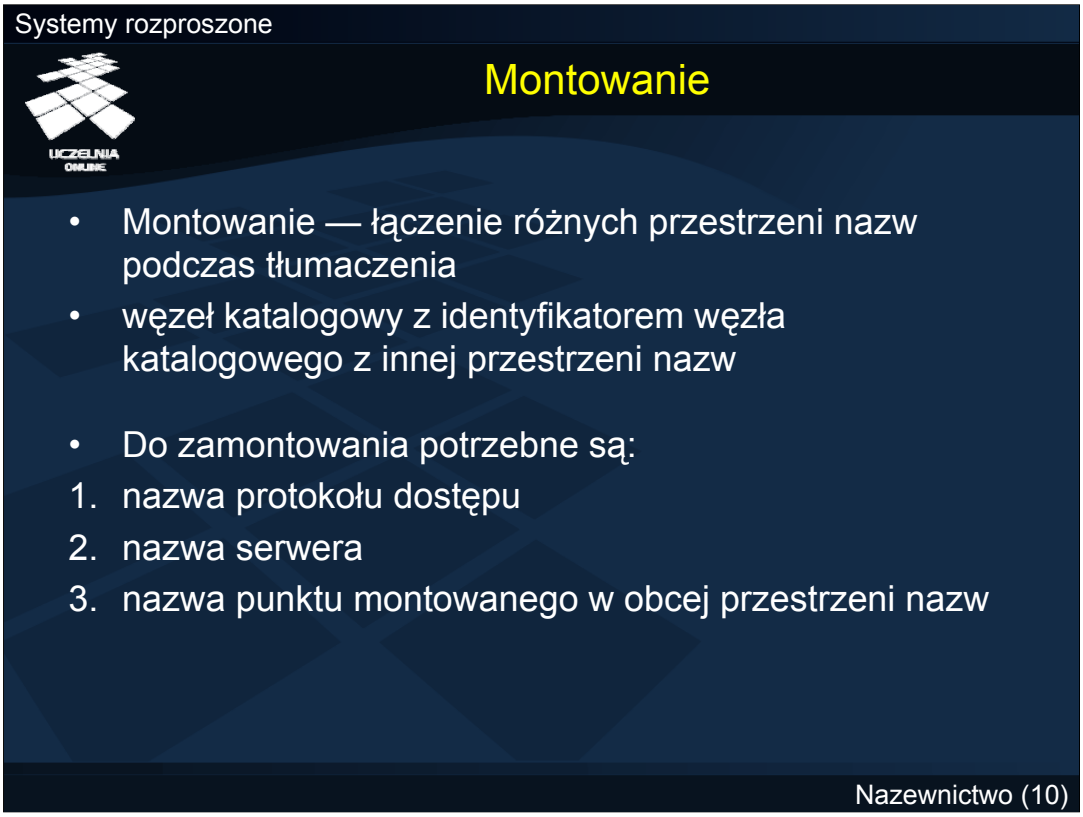


Przestrzeń nazw definiuje organizację danych w postaci drzewa (grafu). Najczęściej wykonywaną operacją na takim drzewie jest operacja wyszukiwania obiektów o określonych nazwach czyli tłumaczenie nazw (inne określenia: rozbiór nazw, rozwiązywanie, rozkładanie, odwzorowywanie). Proces ten najczęściej realizowany jest iteracyjnie, począwszy od korzenia. Przeszukiwane są kolejne tablice katalogowe w celu znalezienia węzła wskazywanego kolejnym członem bezwzględnej nazwy. Pewnym problemem jest jednak znalezienie punktu startowego do rozpoczęcia przeszukiwania struktury katalogowej, ponieważ niskopoziomowy identyfikator korzenia teoretycznie powinien być zapisany w tablicy katalogowej węzła nadrzędnego, a takiego nie ma. W zależności od usługi nazewniczej problem ten różnie się rozwiązuje. Np. w systemach plików jest ogólnie wiadomym gdzie znajduje się blok opisujący katalog główny. Jest to informacja zakodowana w podsystemie obsługi plików bądź zakodowana w inny sposób w systemie operacyjnym.



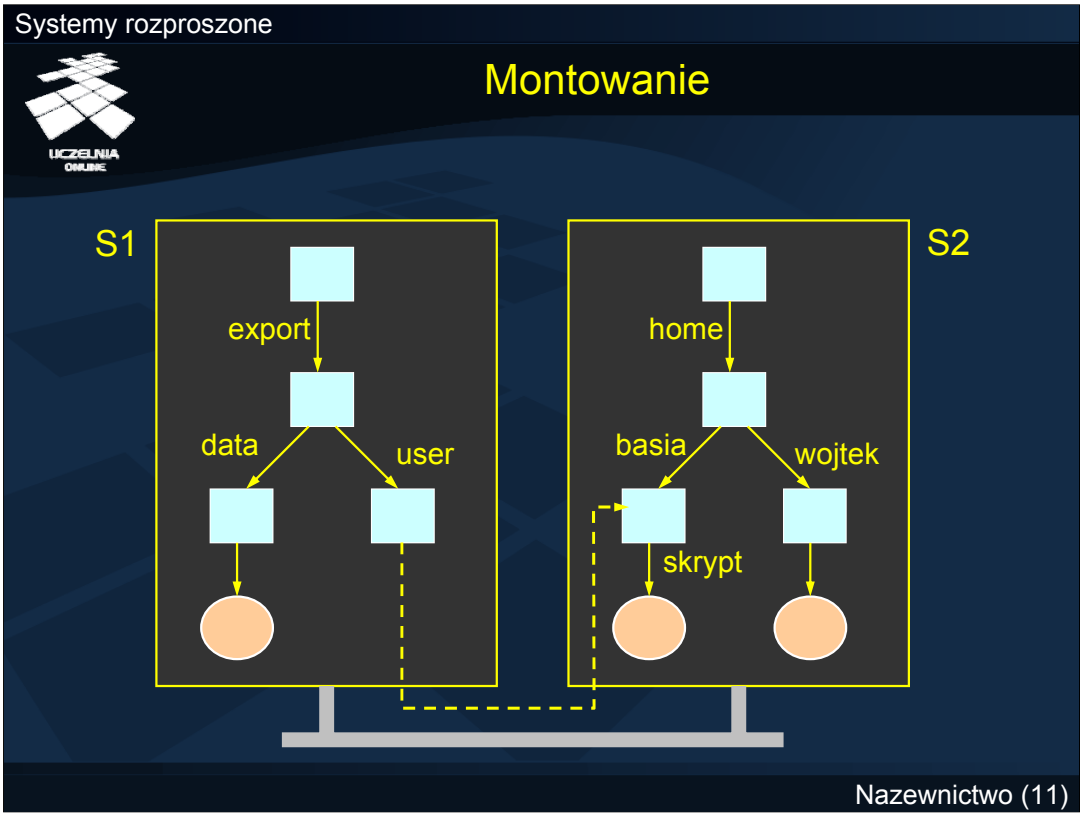
Niektóre systemy nazewnicze dopuszczają istnienie synonimów, zwanych często aliasami (ang. *aliases*). Mechanizm ten jest bardzo użyteczny, gdyż pozwala postrzegać tą samą przestrzeń danych zorganizowaną w różne hierarchie.

Synonimy mogą być implementowane na dwa sposoby. Pierwszy polega na rejestrowaniu kilku równoważnych nazw, w dowolnych miejscach hierarchii nazw, odnoszących się do tych samych zasobów. Rozwiązanie takie jest stosowane np. w uniksowych systemach plików w formie dowiązań twardych (ang. *hard links*). Drugi sposób reprezentacji synonimów polega na utworzeniu specjalnych węzłów-liści w grafie nazewniczym, które nie identyfikują docelowych jednostek, a jedynie zawierają docelowe nazwy, z których należy skorzystać w dalszej kolejności. Proces tłumaczenia nazw po napotkaniu na taki węzeł specjalny inicjuje kolejny proces tłumaczenia dla nazwy pobranej z węzła specjalnego. W uniksowych systemach plików rozwiązanie takie określane jest jako dowiązania symboliczne (ang. *symbolic links*).

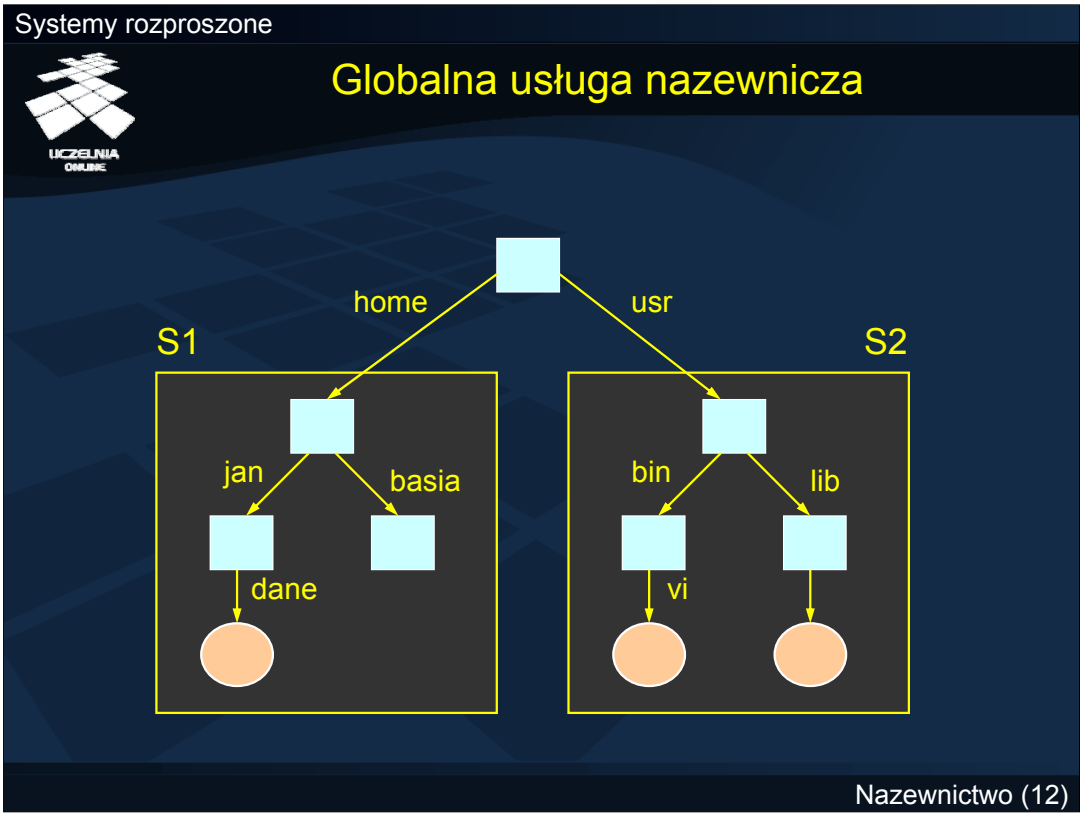


Tłumaczenie nazw można wykorzystać do przezroczystego łączenia różnych przestrzeni nazw. Wystarczy, że w tym celu w wybranym węźle katalogowym pierwszej przestrzeni adresowej umieszczony zostanie identyfikator węzła katalogowego z drugiej przestrzeni nazw. Pierwszy węzeł będzie określany jako **punkt montażu** (ang. *mount point*), a węzeł katalogowy docelowej przestrzeni nazw to **punkt montowany** (ang. *mounting point*). Punkt montowany zazwyczaj jest korzeniem przestrzeni nazw. Podczas tłumaczenia nazwy, która odnosi się do innej przestrzeni nazw następuje przekazanie usłudze nazewniczej tej przestrzeni nazw odpowiedniej części nazwy i proces tłumaczenia jest kontynuowany. Ważne jest aby punkt montażu zawierał pełną informację identyfikującą montowany węzeł, a więc nie tylko lokalny identyfikator, ale także wskazanie na nową przestrzeń nazw.

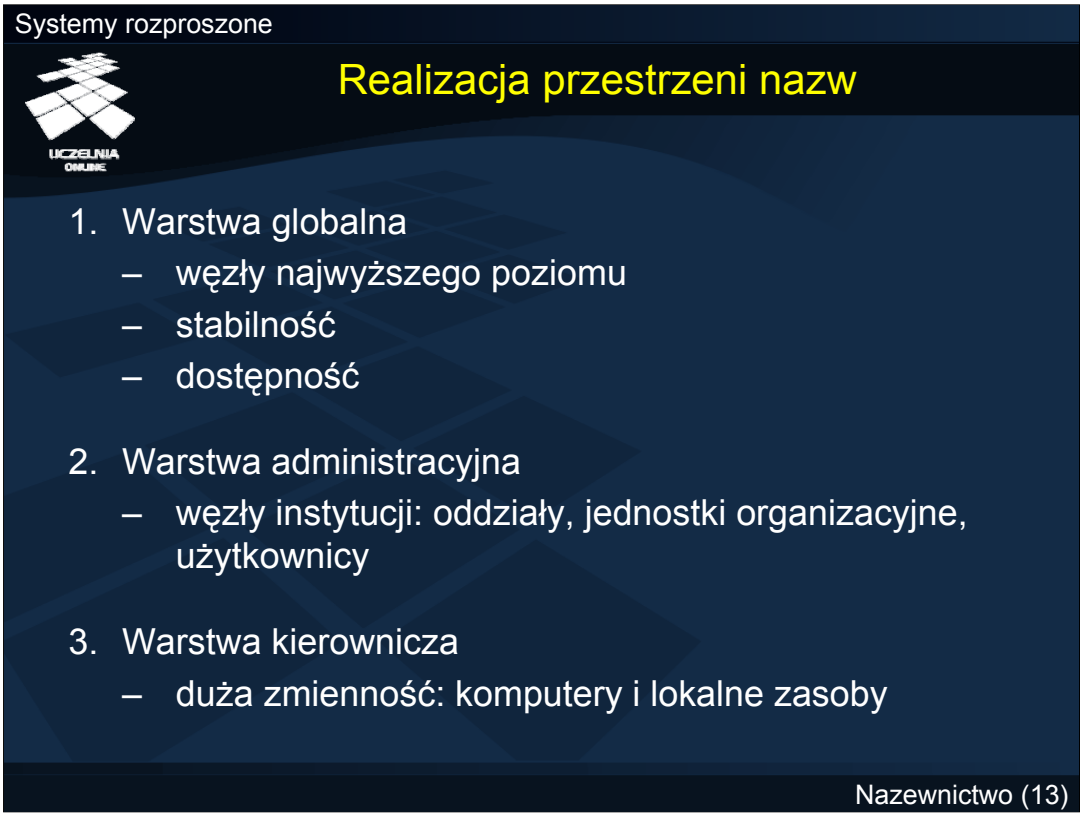
Montowanie wymaga odwzorowania kilku nazw identyfikujących zdalny zasób: protokół musi być skojarzony z odpowiednią implementacją, nazwa serwera musi być przetłumaczona na adres, i w końcu nazwa punktu montowanego na jego identyfikator. Wszystkie parametry identyfikujące zdalny zasób mogą być zakodowane np. w postaci URL. Np. *nfs://galio.cs.put.poznan.pl/usr/local* identyfikuje zdalny węzeł katalogowy */usr/local* na serwerze *galio.cs.put.poznan.pl* dostępny za pośrednictwem protokołu NFS.



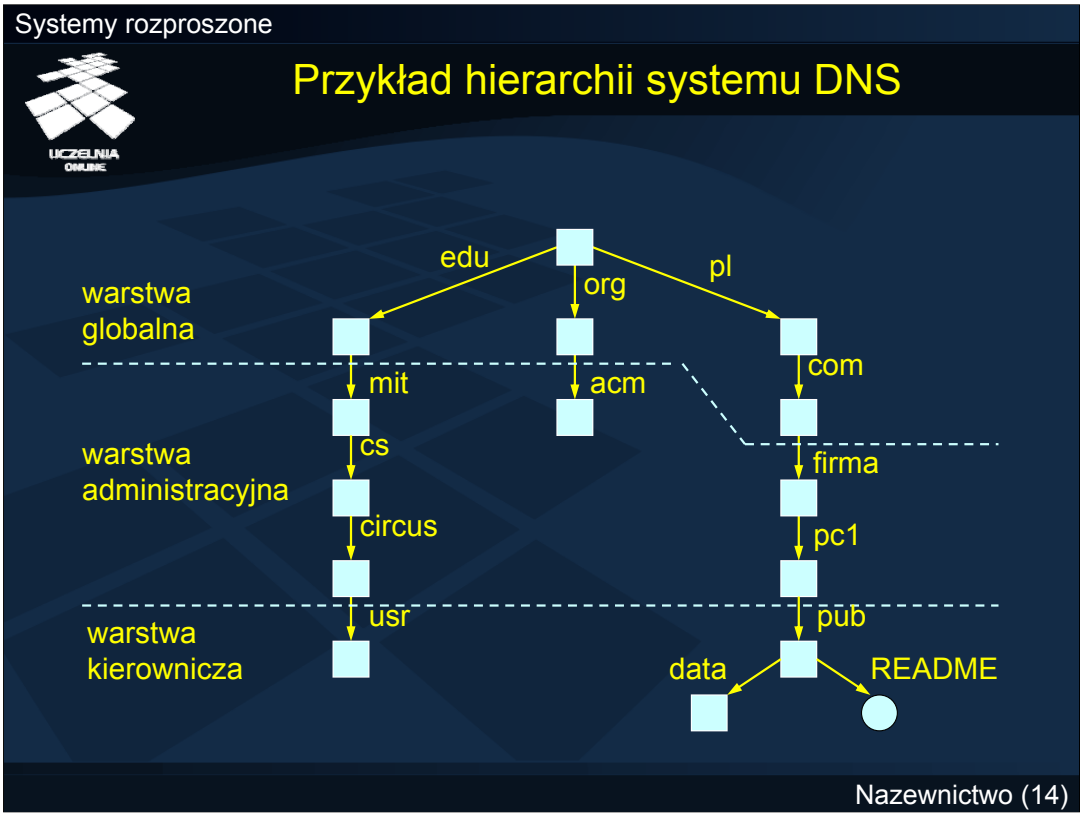
Rysunek pokazuje przykład montowania zdalnego węzła katalogowego z serwera S2 do węzła katalogowego na serwerze S1. Punkt montażu */export/user* zawiera wskazanie na punkt montowany */home/basia* z serwera S2, np. w postaci *nfs://galio.cs.put.poznan.pl/home/basia*. Po zamontowaniu przestrzeni nazw serwera S2 na serwerze S1 można np. posługiwać się nazwą */export/user/skrypt*, która będzie powodować odwołanie do pliku *skrypt* z katalogu */home/basia* na serwerze S2.



Inną metodą integracji kilku przestrzeni nazw jest utworzenie wirtualnego węzła katalogowego, nadrzędnego w stosunku do wszystkich węzłów-korzeni z poszczególnych przestrzeni nazw. Podejście to zastosowano np. w globalnych usługach nazewniczych GNS (ang. *Global Name Service*) firmy DEC. Pewnym problemem tego podejścia jest konieczność zmiany wszystkich używanych dotąd nazw poprzez doklejenie odpowiedniego członu z przodu. Rozwiązaniem może być automatyczne doklejanie prefiksu zgodnego z identyfikatorem serwera do wszystkich nazw i późniejsze tłumaczenie tych prefiksów na pełną ścieżkę. Przykładowo nazwa */jan/dane*, którą posługuje się użytkownik na serwerze S1 zostałaby rozszerzona do *S1:/jan/dane*, a następnie prefiks ten zostałby odwzorowany na nazwę */home* dając ostatecznie */home/jan/dane*. Niestety rozwiązanie to pociąga za sobą konieczność utrzymywania tablicy odwzorowań identyfikatorów serwerów na odpowiednie nazwy, co przy bardzo dużej liczbie serwerów może ograniczać efektywność tego podejścia.



Duże przestrzenie nazw są organizowane najczęściej hierarchicznie. Jeżeli mają obsługiwać również dużą liczbę użytkowników rozproszonych geograficznie, to automatycznie sama usługa nazewnicza również musi być realizowana z wykorzystaniem wielu serwerów. Strukturę dużej przestrzeni nazw można podzielić na trzy logiczne warstwy. Warstwa globalna jest tworzona przez węzły najwyższego poziomu (korzeń i bliskie mu węzły). Na tym poziomie konfiguracja jest raczej stabilna, a najważniejszym wymaganiem stawianym serwerom tej warstwy jest wysoka dostępność. Niżej ulokowana jest warstwa administracyjna, w której ulokowane są węzły reprezentujące poszczególne organizacje i ich oddziały. Na tym poziomie mogą się znaleźć również węzły reprezentujące poszczególnych użytkowników (pracowników). Najniższa warstwa – warstwa kierownicza – zawiera węzły, które są najbardziej podatne na zmiany, a więc węzły reprezentujące poszczególne komputery i zasoby przechowywane na tych komputerach. Warstwa ta jest o tyle szczególna, że jej zarządzaniem zajmują się również indywidualni użytkownicy, modyfikując wpisy za które są odpowiedzialni.



Przestrzeń nazw systemu DNS (ang. *Domain Name Service*) podzielona jest na **strefy** (ang. *zones*). Każda strefa jest obsługiwana przez oddzielny serwer nazw. Wymaga się, aby każda strefa miała przynajmniej jeden serwer zapasowy, który jest repliką serwera podstawowego. Serwery obsługujące nazwy wyższych poziomów mają większą liczbę kopii dla zwiększenia ich dostępności, gdyż awaria takiego serwera mogłaby pociągać za sobą niemożliwość przetłumaczenia nazw pochodzących z dużej części przestrzeni nazw.

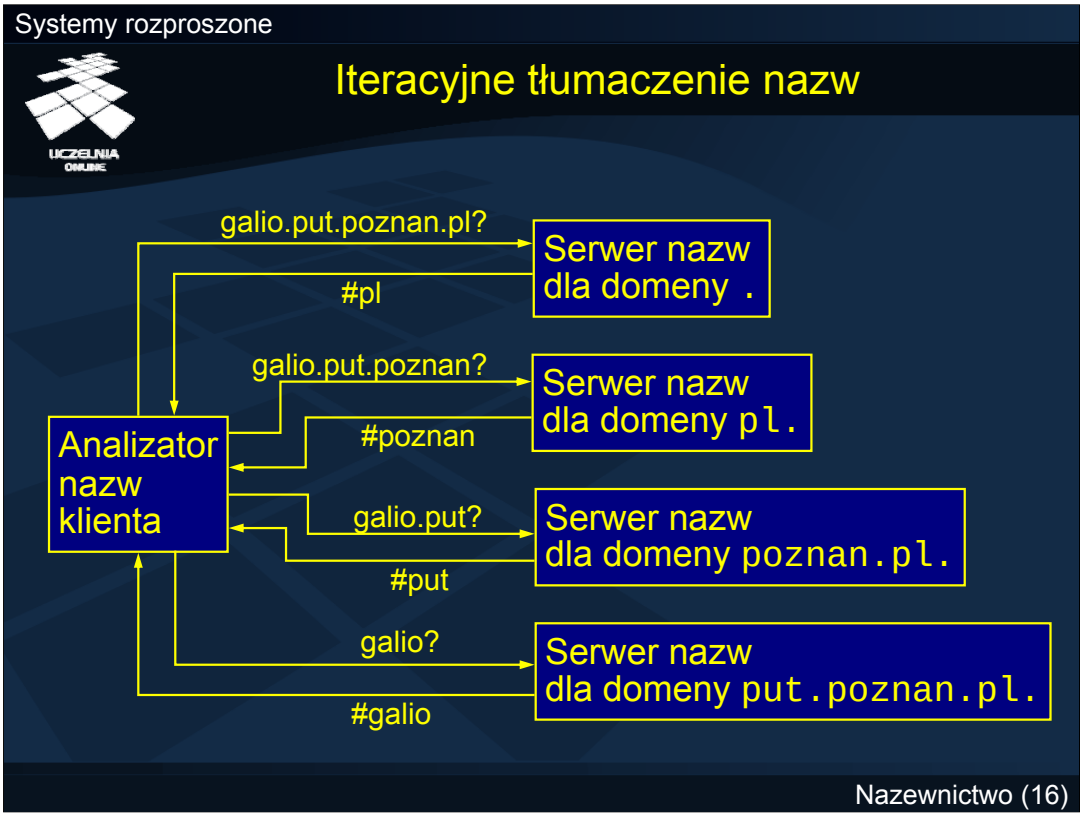
Efektywność dostępu do danych zapewnia się w dużej mierze poprzez stosowanie pamięci podręcznych we wszystkich serwerach nazw. Jest to szczególnie istotne w odniesieniu do serwerów obsługujących główne strefy lub strefy znajdujące się wysoko w hierarchii. Do serwerów takich potencjalnie kierowanych może być bardzo wiele zapytań ze względu na dużą liczbę podlegających im nazw. Ze względu jednak na dużą stabilność odwzorowań znajdujących się wysoko w hierarchii, dane te mogą być przez długi czas buforowane w innych serwerach niższego poziomu. W ten sposób redukowana jest liczba zapytań kierowanych do głównych serwerów.

Awarie serwerów nazw w warstwie kierowniczej są odczuwalne głównie przez użytkowników znajdujących się w danej jednostce organizacyjnej i ewentualnie ich klientów. W tym przypadku nacisk kładzie się raczej na efektywność pracy takiego serwera niż na jego dostępność, która jest gwarantowana przez lokalnego administratora. Efektywność należy tu jednak uzyskiwać głównie poprzez instalację oprogramowania na szybkim komputerze i zapewnienie dobrej łączności, gdyż intensywne stosowanie pamięci podręcznych w przypadku często zmieniających się danych może nie dać dobrych rezultatów.

Serwery nazw DNS

Warstwa	Globalna	Administracyjna	Kierownicza
Skala geograficzna	światowa	instytucja	oddział
Liczba węzłów	mała	duża	olbrzymia
Czas przeszukiwania	sekundy	milisekundy	natychmiast
Propagacja uaktualnień	leniwa	natychmiast	natychmiast
Liczba kopii	wiele	mało albo wcale	wcale
Przechowywanie podręczne	tak	tak	czasami

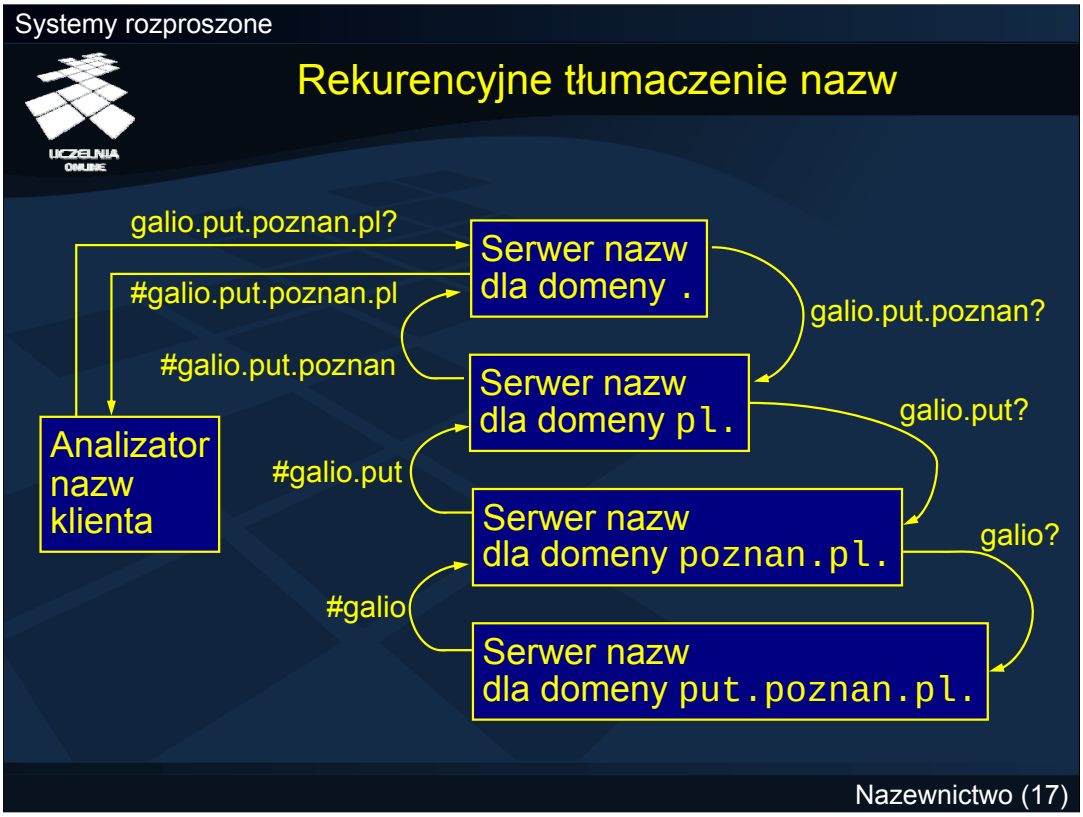
Tabela przedstawia własności serwerów nazw pracujących w różnych warstwach przestrzeni nazw w systemie DNS. W warstwie globalnej intensywnie stosowane jest zwielokrotnianie i przechowywanie podręczne. Działanie to jest konieczne dla uzyskania wysokiej dostępności danych. Oczywiście generuje to problemy z utrzymaniem danych w stanie spójnym. Tym bardziej, że kopie danych rozrzucone są po całej sieci (głównie w postaci zawartości pamięci podręcznych). Modyfikacja takich danych zawsze rozciągnięta jest mocno w czasie, chociażby ze względu na opóźnienia komunikacyjne nieuniknione w sieci tej skali.



Rozproszenie przestrzeni nazw na wiele serwerów przekłada się bezpośrednio na sposób realizacji procesu tłumaczenia nazw, który musi angażować wiele serwerów. Tłumaczenie nazw w systemie DNS może być realizowane na dwa sposoby: **iteracyjnie** (ang. *iterative name resolution*) lub **rekurencyjnie** (ang. *recursive name resolution*). Poniższy opis metod odpytywania systemu DNS zakłada dla uproszczenia, że nie jest wykorzystywane zwielokrotnianie danych ani przechowywanie podręczne.

Tłumaczenie nazw realizowane jest przez oprogramowanie klienta zwane **analizatorem nazw** (ang. *name resolver*). Tłumaczenie iteracyjne, przedstawione na slajdzie, kontrolowane jest w całym zakresie przez klienta. Rozpoczyna się ono od wysłania zapytania o nazwę do serwera domeny głównej. W tym przypadku realizowane jest tłumaczenie nazwy *galio.put.poznan.pl*. Serwer domeny głównej nie zna odpowiedzi na to pytanie i nie może dokonać pełnego przetłumaczenia tej nazwy na adres, ale wie jaki serwer jest odpowiedzialny za domenę *pl*, która występuje na końcu tłumaczonej nazwy. W odpowiedzi serwer domeny głównej wysyła adres serwera domeny *pl* (oznaczony jako #pl). Klient uzyskując tą informację, wysyła kolejne zapytanie tym razem do serwera domeny *pl*, próbując przetłumaczyć pozostałą część poszukiwanej nazwy, a więc *galio.put.poznan*. Serwer nazw domeny *pl* odpowiada wskazując na serwer nazw domeny *poznan.pl*. Ostatecznie pytanie trafia do serwera nazw domeny *put.poznan.pl*, gdzie przechowywane są dane o komputerach znajdujących się w tej domenie, a więc m.in. o komputerze *galio*. Klient uzyskuje w ten sposób adres odpowiadający nazwie *galio.put.poznan.pl*.

Zakłada się, że klient zna adres serwera domeny głównej. W praktyce lista tych serwerów nie jest zbyt obszerna i jest aktualizowana stosunkowo rzadko, więc jej utrzymywanie nie jest problematyczne.



Rekurencyjne tłumaczenie nazw polega na zleceniu translacji nazwy pojedynczemu serwerowi. Serwer domeny głównej nie znając odpowiedzi na pytanie przekazuje je dalej, nie odsyłając odpowiedzi od razu klientowi. Zapytanie przekazywane jest do serwera, który powinien znać odpowiedź lub takiego, który będzie w stanie przeprowadzić dalszy proces tłumaczenia nazwy. W przedstawionym przykładzie serwer domeny głównej przekazuje zapytanie do serwera domeny *pl*, ten do serwera domeny *poznan.pl*, a ten w końcu do serwera domeny *put.poznan.pl*, gdzie następuje ostateczne przetłumaczenie nazwy na adres. Odpowiedź wraca tą samą drogą, ale w przeciwnym kierunku.

Z punktu widzenia klienta, odpowiedź zawsze nadchodzi z serwera, który został odpytany, co jest wygodne, gdyż upraszcza oprogramowanie klienta. Wadą tego podejścia jest większe zaangażowanie poszczególnych serwerów w proces tłumaczenia nazw, co pociąga za sobą ich większe obciążenie. W praktyce więc serwery domeny głównej nie realizują zapytań rekurencyjnych. Inną zaletą odpytywania rekurencyjnego jest oszczędność komunikacji. Odpytywanie odległych (w sensie geograficznym) serwerów nazw w trybie iteracyjnym pociąga za sobą konieczność przesyłania komunikatów na duże odległości (przez wiele routerów). W trybie rekurencyjnym informacja przekazywana jest na mniejsze odległości.

Przesyłanie odpowiedzi wykorzystywane jest do wypełniania pamięci podręcznych poszczególnych serwerów. Przykładowo serwer domeny *pl* po wykonaniu powyższego zapytania uzyskuje informację o adresie serwera nazw dla domeny *put.poznan.pl*, którą to informację może wykorzystać w przyszłości dla zoptymalizowania następnego pytania o inny komputer z tej domeny. W ten sposób postępuje proces *uczenia się* serwerów, w którym dzięki przechowywaniu podręcznemu serwery są w stanie bezpośrednio odpowiedzieć na najczęściej zadawane pytania.

W praktyce zapytania zgłaszane przez klientów nie są wysyłane bezpośrednio do serwera domeny głównej, ale do lokalnego serwera nazw, którego głównym zadaniem jest pośredniczenie w tłumaczeniu nazw. Serwer taki ma za zadanie odpytywanie innych serwerów w celu uzyskania odpowiedzi i następnie przechowuje podręcznie uzyskane wyniki. Ponieważ serwer taki znajduje się blisko klienta, dostęp do niego jest bardzo szybki. Po pewnym czasie serwer taki zna również odpowiedzi na dużą część pytań klientów lub przynajmniej zna serwery nazw głównych domen, co pozwoli w przyszłości szybko uzyskać właściwą odpowiedź. Z drugiej strony uruchomienie lokalnego serwera zwalnia analizator nazw klienta z konieczności przechowywania listy adresów serwerów nazw domeny głównej. Wystarczy w celu odwzorowania nazwy posłużyć się adresem lokalnego serwera nazw.

Rekordy zasobowe systemu DNS

Typ rekordu	Opis
SOA	Informacja o całej strefie
A	Adres komputera
MX	Adres serwera pocztowego dla domeny
NS	Adres serwera nazw dla domeny
CNAME	Dodatkowa nazwa (alias) dla węzła
PTR	Nazwa serwera o wskazanym adresie IP

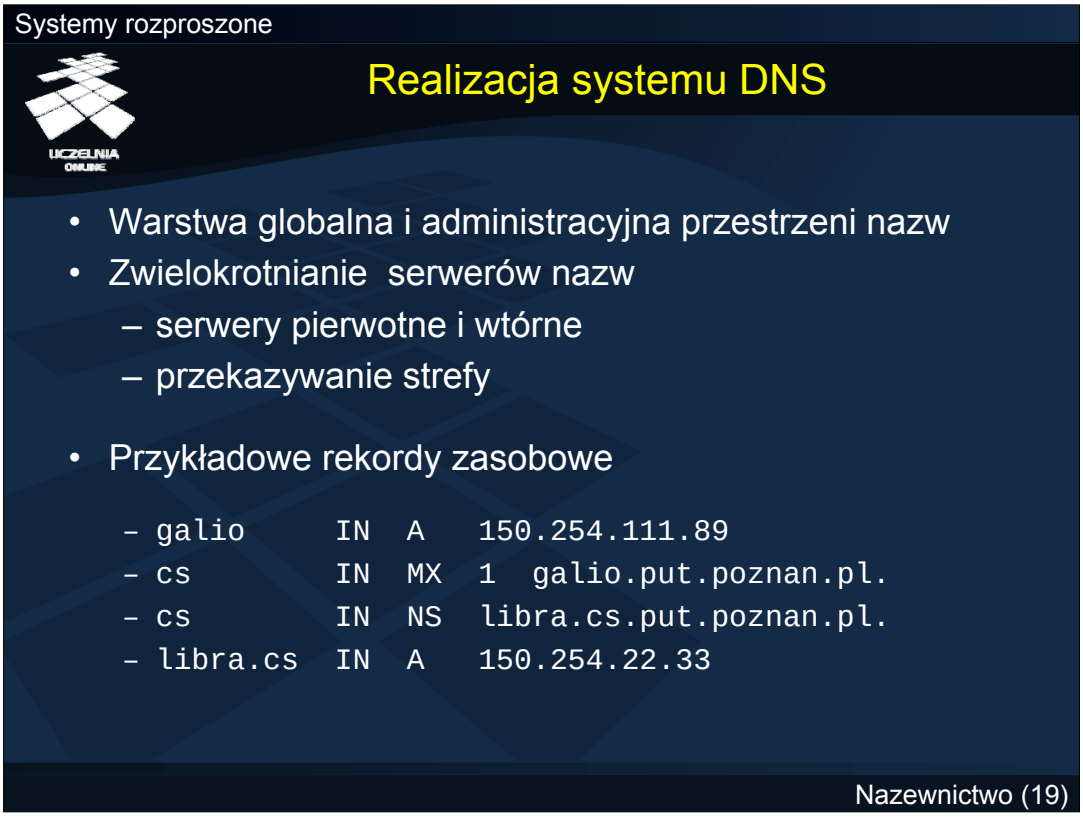
Przestrzeń nazw w systemie DNS jest zorganizowana w postaci drzewa. Nazwy poszczególnych węzłów mają postać sekwencji etykiet rozdzielonych kropką, począwszy od etykiety najniższej w hierarchii a skończywszy na etykiecie najwyższej w hierarchii. Ponieważ hierarchia jest drzewem, to nazwy krawędzi tego drzewa wykorzystuje się do etykietowania samych węzłów. Korzeń jest oznaczany po prostu „.”, choć w zapisie nazw węzłów jest to z reguły pomijane. Nazwy nie rozróżniają małych i wielkich liter. Przykładowa nazwa *put.poznan.pl.* oznacza węzeł *put* będący podrzędnym węzłem w stosunku do węzła *poznan*, który z kolei jest podrzędny w stosunku do *pl*. Węzeł *pl* podlega bezpośrednio korzeniowi hierarchii.

Poddrzewa przestrzeni nazw w systemie DNS są nazywane **domenami** (ang. *domain*). Nazwa ścieżki od korzenia całej przestrzeni nazw do korzenia domeny nazywamy **nazwą domeny** (ang. *domain name*). Domena może obejmować kilka stref, np. domena *poznan.pl* może obejmować zarówno strefę *put.poznan.pl* jak i strefę *man.poznan.pl*.

Konfiguracja domeny opisana jest zbiorem **rekordów zasobowych** (ang. *resource records*). Tabelka przedstawia najczęściej wykorzystywane typy rekordów zasobowych. Rekord SOA (ang. *Start of Authority*) zawiera podstawowe informacje opisujące strefę, m.in. adres poczty elektronicznej administratora systemu. Rekord A (adres), najczęściej stosowany, zawiera adres IP przypisany do konkretnej nazwy domenowej. Dla komputerów z kilkoma adresami IP definiuje się kilka rekordów A. Rekord MX (ang. *mail exchange*) zawiera wskazanie na serwer obsługujący pocztę dla danej domeny. Dla strefy może być zdefiniowanych kilka rekordów MX, każdy kolejny wskazuje na zapasowe serwery pocztowe. Rekordy NS (ang. *name server*) zawierają adres serwera nazw obsługującego daną strefę. Na podstawie rekordów NS serwery nazw wiedzą gdzie kierować zapytania w celu dalszego tłumaczenia nazwy, której dany serwer nie potrafi do końca przetłumaczyć.

Rekord CNAME (ang. *canonical name* – nazwa kanoniczna) pozwala na zdefiniowanie dodatkowych nazw dla komputerów. W rekordzie tym znajduje się wskazanie na **nazwę kanoniczną** komputera, czyli jego unikalną, podstawową nazwę. Mechanizm ten wykorzystuje się często do definiowania pomocniczych nazw *www* czy *ftp* w ramach domen, które reprezentują konkretne komputery z tych domen. Np. nazwa *www.put.poznan.pl* może być dodatkową nazwą dla komputera *libra.put.poznan.pl*, albo inaczej mówiąc *libra.put.poznan.pl* jest kanoniczną nazwą dla nazwy *www.put.poznan.pl*.

System DNS utrzymuje również informacje o odwzorowaniach odwrotnych, a więc o tłumaczeniach adresów IP na nazwy. Do reprezentowania tej informacji służą rekordy PTR (ang. *pointer* – wskaźnik). Odwzorowania odwrotne przechowywane są w specjalnej domenie *in-addr.arpa*, gdzie poszczególne nazwy budowane są na podstawie adresów IP, poprzez odwrócenie kolejności poszczególnych bajtów składowych adresu. Np. adres 150.254.30.34 reprezentowany jest nazwą *34.30.254.150.in-addr.arpa*. W rekordzie PTR znajduje się nazwa kanoniczna przypisana do danego adresu IP.



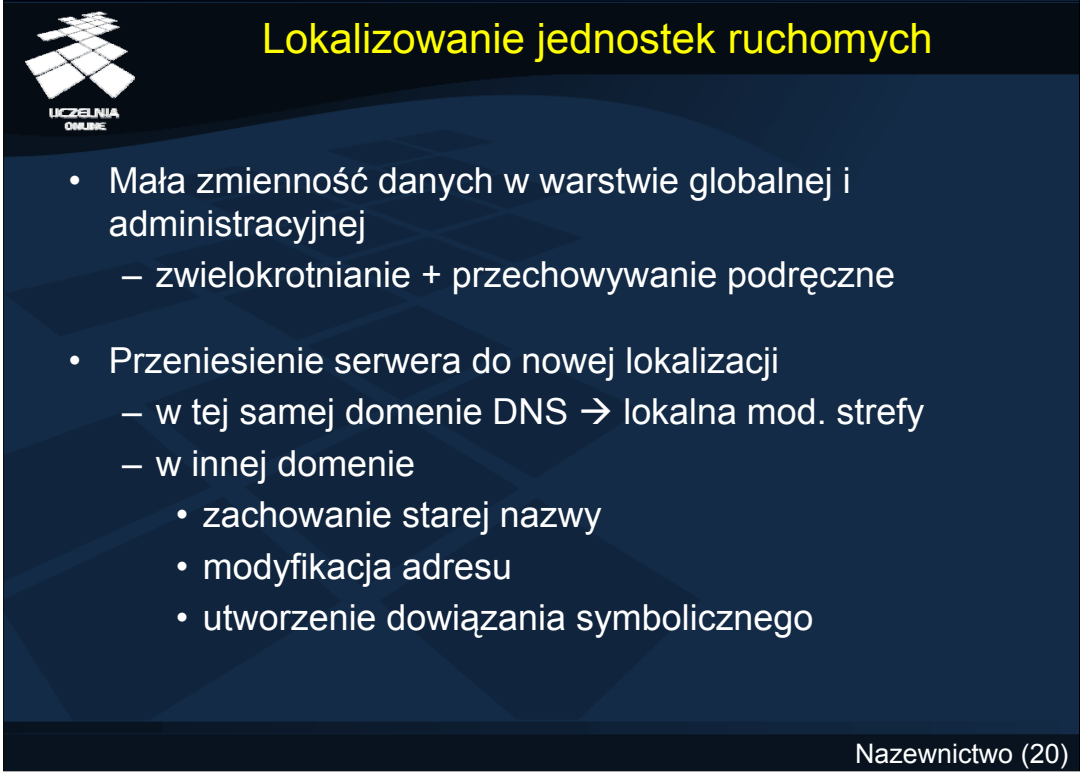
System DNS realizuje zasadniczo zadania warstwy globalnej i administracyjnej. Warstwa kierownicza realizowana jest przez system plików lub przez inne usługi.

Każda strefa jest obsługiwana przez serwer nazw, który z reguły jest zwielokrotniany dla zwiększenia dostępności. Synchronizacja danych pomiędzy serwerem **pierwotnym** (ang. *primary*) a **wtórnym** (ang. *secondary*) odbywa się poprzez okresową weryfikację aktualności danych na podstawie numeru wersji pliku strefowego (tekstowej bazy danych zawierającej rekordy zasobowe strefy). Po stwierdzeniu aktualizacji serwer wtórny prosi o przesłanie aktualnej konfiguracji, co określane jest jako **przekazywanie strefy** (ang. *zone transfer*).

W slajdzie przedstawiono przykładowe rekordy zasobowe jakie mogłyby się znaleźć w pliku strefowym dla strefy *put.poznan.pl*. Pierwszy rekord A definiuje nazwę *galio* w domenie *put.poznan.pl*, a więc nazwę *galio.put.poznan.pl*, której przypisany został adres 150.254.111.89. Rekord MX wskazuje na serwer pocztowy dla poddomeny *cs.put.poznan.pl*. Kolejny rekord (NS) definiuje serwer nazw dla poddomeny *cs*; będzie nim komputer *libra.cs.put.poznan.pl*. Ponieważ sama nazwa serwera nazw dla domeny podrzędnej nie jest wystarczająca dla skontaktowania się z tym serwerem, w pliku znajduje się również tzw. rekord łączący, zawierający adres komputera *libra* (rekord A). Dzięki tym dwóm ostatnim rekordom możliwe jest poprawne przekazanie pytań o nazwy z domeny *cs.put.poznan.pl* do odpowiedniego serwera nazw.

W oddzielnym pliku strefowym zapisywane są odwzorowania odwrotne (z domeny *in-addr.arpa*). W pliku tym może się znaleźć rekord PTR tłumaczący adres IP na nazwę. Dla powyższego przykładu domena odwrotna będzie miała nazwę *111.254.150.in-addr.arpa*, a odpowiedni rekord PTR będzie miał postać:

```
89  IN  PTR  galio.put.poznan.pl.
```



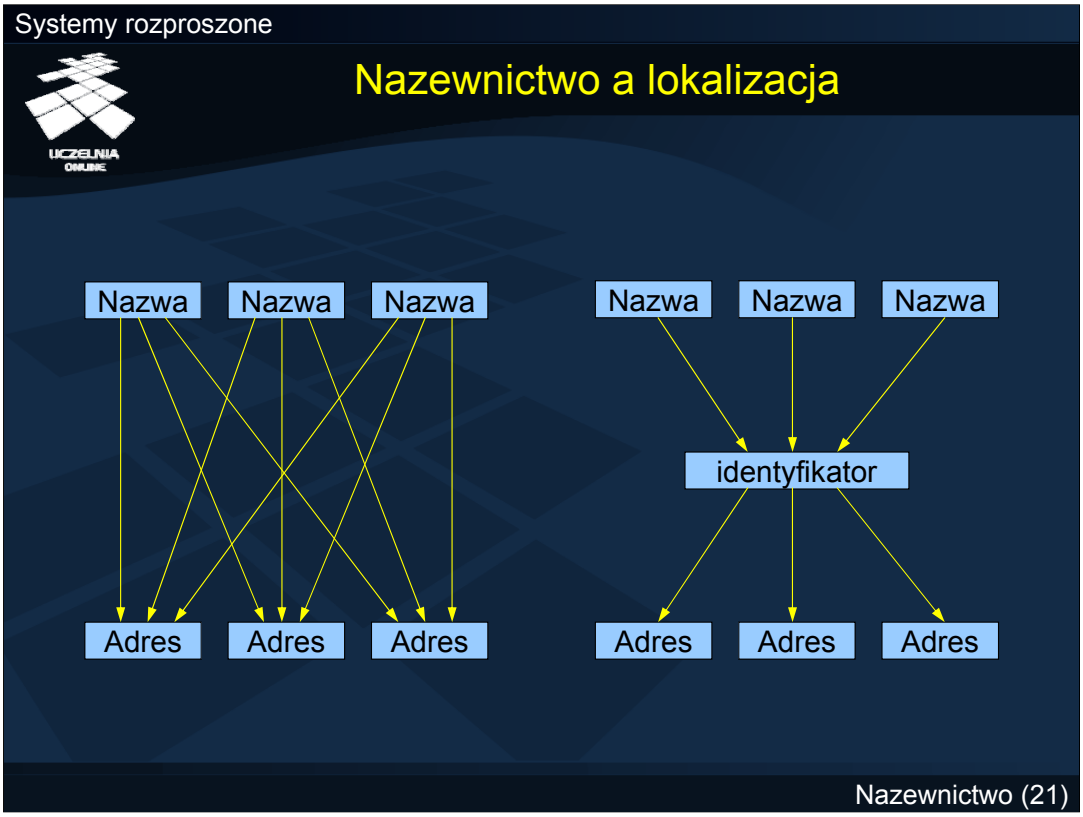
Tradycyjne systemy nazewnicze (takie jak np. system DNS) nie nadają się do zarządzania często zmieniającą się informacją. Sytuacja taka występuje w przypadku realizacji odwzorowań jednostek mobilnych, zmieniających swoje adresy. Założenie o stabilności odwzorowań pozwala w przypadku systemu DNS stosować na szeroką skalę zwielokrotnianie i przechowywanie podręczne, dlatego, że ryzyko niespójności danych jest minimalne.

Zmiana lokalizacji komputera może dotyczyć zmiany adresu przy niezmienionej nazwie. Ta zmiana może być zrealizowana lokalnie przez administratora poprzez modyfikację odpowiedniego rekordu zasobowego. Propagacja takiej zmiany będzie w tym przypadku szybka.

Zmiana lokalizacji komputera wraz ze zmianą jego nazwy jest już większym problemem. Z reguły użytkownicy spodziewają się, że będą mogli korzystać ze starej nazwy bez zmian. Zakłada się więc, że nazwa będzie pełniła rolę identyfikatora, a więc niezmiennej, unikalnej nazwy. Nazwa w systemie DNS nie spełnia jednak tego wymagania – zmienia się wraz ze zmianą lokalizacji, a stara nazwa może być powtórnie wykorzystana.

Przeprowadzenie zmiany nazwy serwera z zachowaniem starej można zrealizować na dwa sposoby. Pierwszy polega na zapisaniu pod starą nazwą nowego adresu. Niestety kolejne zmiany lokalizacji będą wymagały dokonywania modyfikacji zapisów na zdalnym, być może nie kontrolowanym już, serwerze nazw. W efekcie dalsze zmiany będą zbyt wolno propagowane.

Drugie rozwiązanie polega na zmianie starego wpisu, tak aby była dowiązaniem symbolicznym do nowej nazwy. Efektem ubocznym tego rozwiązania jest wydłużenie czasu realizacji tłumaczenia nazwy, ponieważ tłumaczenie to będzie musiało teraz dotyczyć dwóch nazw (starej i nowej). Przy dalszym przemieszczaniu liczba kroków tłumaczenia może jeszcze wzrosnąć.



Tradycyjne usługi nazewnicze dokonują odwzorowania nazwy bezpośrednio na adres. Każda zmiana adresu wymusza automatycznie konieczność zmiany odwzorowania nazwa-adres (rysunek po lewej). Lepszym rozwiązaniem jest wprowadzenie identyfikatorów jednostek (rysunek po prawej). Identyfikatory nigdy się nie zmieniają i nie mogą być użyte ponownie do identyfikacji innych jednostek. Identyfikatory nie muszą mieć postaci wygodnej do użycia dla człowieka; z reguły mają postać binarną.

Usługi nazewnicze zwracają identyfikator jednostki. Ponieważ identyfikator nigdy się nie zmienia można go lokalnie zapamiętać i dowolnie długo przechowywać. Nazwa danej jednostki może się w przyszłości zmienić, ale to nie wymaga ponownego odpytywania usługi nazewniczej, ponieważ znany jest już identyfikator poszukiwanej jednostki. Odwołanie się do jednostki wymaga jednaj jej zlokalizowania i jest to zadanie **usługi lokalizacji** (ang. *location service*). Usługa lokalizacji dokonuje odwzorowania identyfikatora jednostki na adres (lub zbiór adresów).

Lokalizacja — proste rozwiązania

1. Rozgłaszanie
   - np. protokół ARP (ang. *Address Resolution Protocol*)
   - problem skalowalności
   - rozsyłanie (np. lokalizacja komputerów w firmie)
   - lokalizacja najbliższej kopii
2. Wskaźniki naprowadzające (ang. *forward pointers*)
   - początkowa lokalizacja z usługi nazewniczej
   - wydłużanie się łańcucha
   - konieczność przechowywanie wskaźników
   - ryzyko przerwania łańcucha

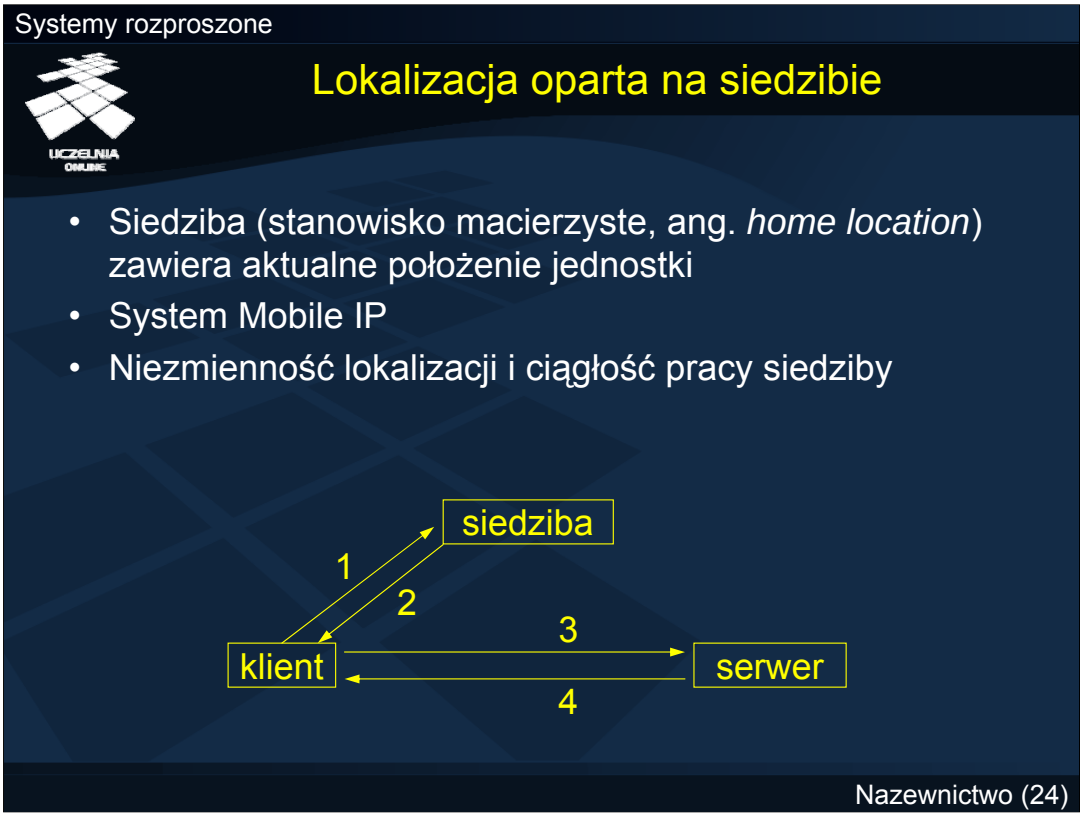
Lokalizację jednostek można przeprowadzić z wykorzystaniem **rozgłaszania**, o ile warstwa komunikacyjna umożliwia wykonywanie takiej operacji. Rozgłoszona wiadomość dociera do wszystkich jednostek, a odpowiada tylko ta, która spełnia określone w rozgłoszeniu warunki. Przykładem zastosowania tego podejścia jest protokół ARP (ang. *Address Resolution Protocol*) mający na celu uzyskanie niskopoziomowego adresu komputera (adres z warstwy łącza danych) na podstawie jego adresu IP. Adres IP wysyłany jest w komunikacie rozgłoszeniowym i odpowiedź wysyłana jest tylko przez komputer, który faktycznie korzysta z tego adresu.

Rozgłaszanie staje się nieefektywne w przypadku dużych sieci, ponieważ angażuje wszystkie komputery i blokuje na pewien czas możliwość komunikacji wszystkich komputerów. Pewnym rozwiązaniem tego problemu jest zastosowanie częściowego rozgłaszania czyli **rozsyłania** (ang. *multicast*). Rozesłanie wiadomości oznacza wysłanie jej do określonej grupy odbiorców. Istnieją protokoły w sieci Internet umożliwiające rozsyłanie wiadomości w skali całego globu. Mechanizm ten można wykorzystać do lokalizacji komputerów firmowych, które dołączając się do sieci uzyskują dynamicznie adresy IP. Każdy z komputerów po uzyskaniu dostępu do sieci może dołączyć się do odpowiedniej grupy rozgłoszeniowej, co umożliwi późniejsze wysłanie do tej grupy pytania lokalizacyjnego.

Mechanizm rozgłaszania/rozsyłania można wykorzystać również do lokalizacji najbliższej kopii danych. Odpowiedź na komunikat rozgłoszeniowy wysyłana będzie w tym przypadku przez wiele serwerów, ale istotna będzie kolejność odbioru tych wiadomości. Można założyć, że komputer który odpowie jako pierwszy jest najbliższy (lub najmniej obciążony).

Innym prostym rozwiązaniem problemu lokalizacji są **wskaźniki naprowadzające** (ang. *forward pointers*). Mechanizm ten polega na pozostawianiu w starej lokalizacji wskaźnika do nowej lokalizacji. Poszukiwanie jednostki sprowadza się wtedy do przejścia całego łańcucha wskaźników, aż dotrze się do właściwej jednostki. Rozwiązanie to jest bardzo proste koncepcyjnie i realizacyjnie, ale ma kilka wad. Łańcuch wskaźników może się bardzo wydłużyć, co spowoduje nieakceptowalne wydłużenie czasu oczekiwania na odpowiedź. Inna wada to konieczność przechowywania wskaźników w poszczególnych węzłach przez cały czas funkcjonowania systemu. W końcu rozwiązanie to może, w przypadku utraty któregoś ze wskaźników, doprowadzić do nieosiągalności jednostki.

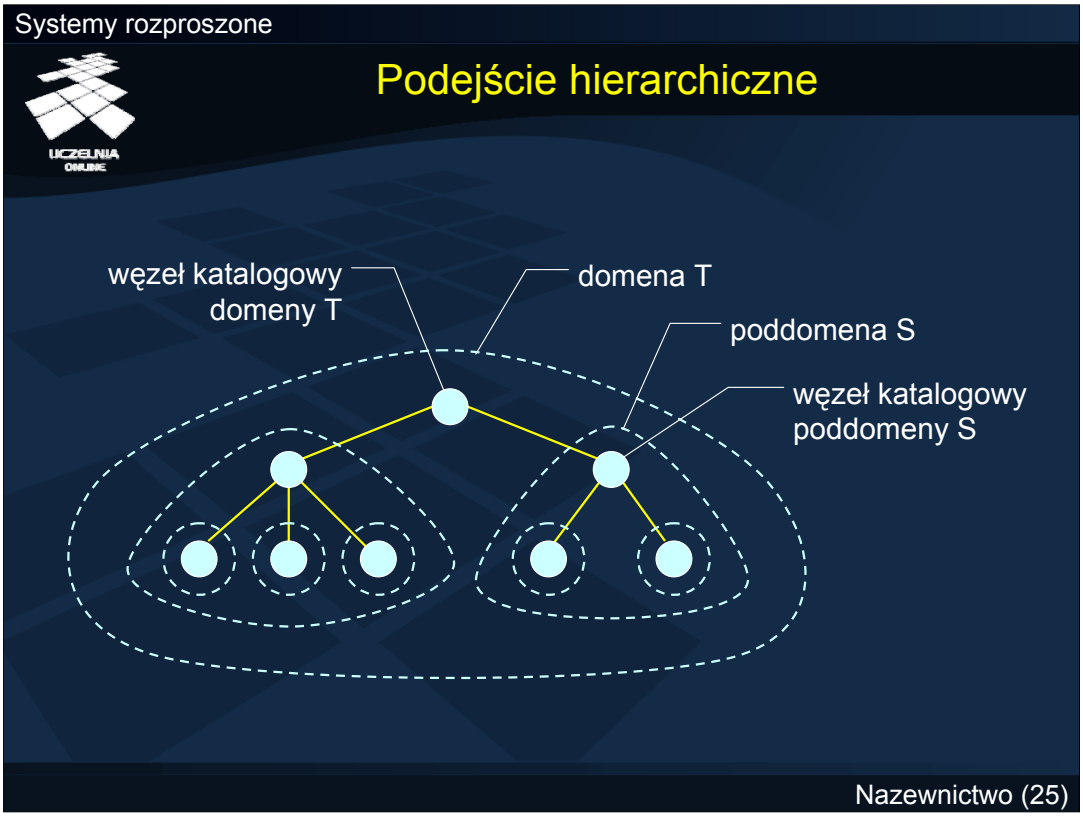
Rysunek przedstawia działanie wskaźników naprowadzających. Żądanie klienta trafia (na podstawie informacji z usługi nazewniczej) do serwera S1, gdzie znajduje się wskaźnik zawierający adres serwera S2. Żądanie trafia do serwera S2, gdzie znajduje się wskaźnik do serwera S3. W serwerze S3 jest poszukiwany obiekt i zlecenie może zostać zrealizowane. Odpowiedź z serwera S3 może trafić bezpośrednio do serwera inicjującego przetwarzanie (S1) lub przejść tą samą drogę w odwrotnym kierunku po to, aby zaktualizować wskaźniki. W pierwszym podejściu odpowiedź trafi szybciej do klienta. W drugim kolejne żądanie kierowane do S1 po aktualizacji wskaźników będzie już mogło trafić bezpośrednio do serwera S3, bez względu na długość poprzedniej ścieżki. Działanie takie powoduje więc szybkie skracanie ścieżek przekierowań pomiędzy serwerami.



Metoda lokalizacji oparta na wskaźnikach naprowadzających może doprowadzić do niedostępności zasobu jeżeli któryś ze wskaźników na ścieżce przestanie być dostępny. W takich sytuacjach można się wspierać (jako metodą awaryjną) wprowadzeniem **siedziby** (stanowiska macierzystego, ang. *home location*) dla każdej jednostki. Jest to miejsce, pod którym zawsze można uzyskać informację o aktualnej lokalizacji jednostki.

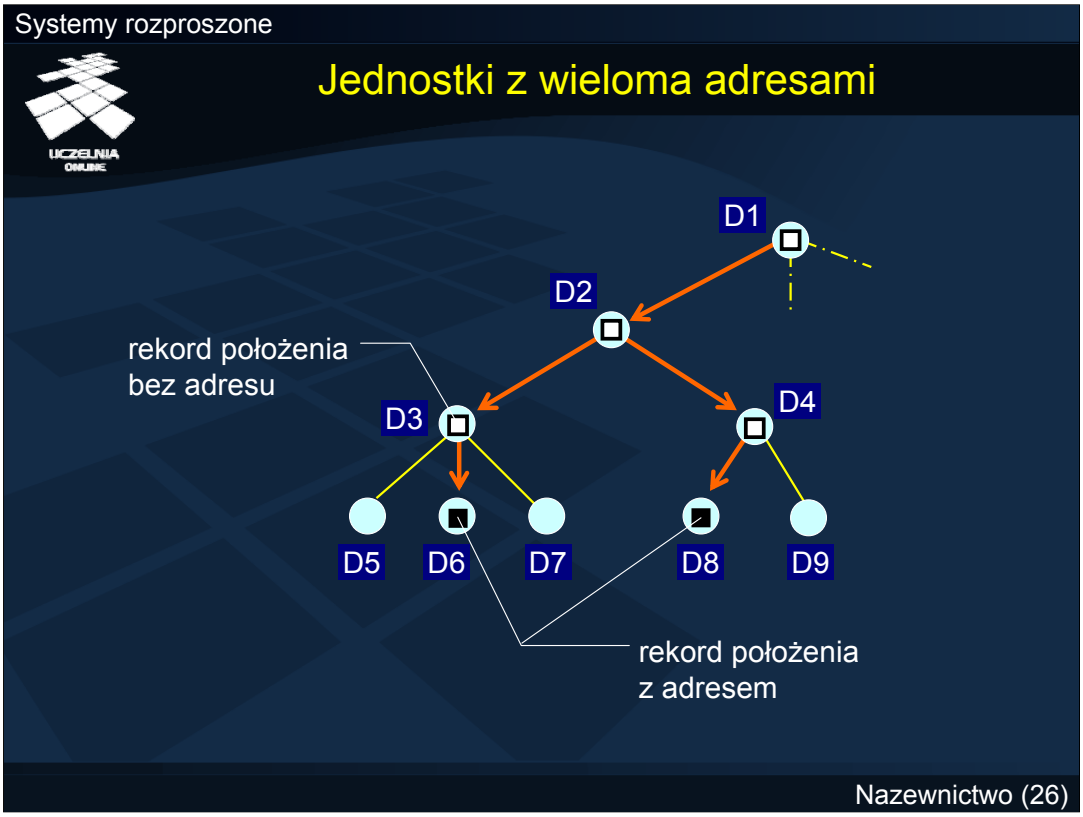
Rozwiązanie oparte na siedzibie zastosowano w systemie Mobile IP umożliwiającym komunikację z przemieszczającymi się komputerami. Dla każdego mobilnego komputera przewidziano program komunikacyjny (agent siedziby), dostępny zawsze pod tym samym adresem IP. Klient komunikuje się najpierw z agentem (1), próbując uzyskać aktualny adres serwera. Agent odpowiada (2) wysyłając adres serwera. Dalsza komunikacja (3-4) odbywa się już bezpośrednio pomiędzy klientem a serwerem. Wadą tego rozwiązania w przypadku stosowania w sieciach na szeroką skalę jest konieczność przeprowadzenia dodatkowej wymiany komunikatów z siedzibą, która może być zlokalizowana w zupełnie innym miejscu niż właściwy serwer.

Inną wadą metody opartej na siedzibie jest konieczność zapewnienia niezmiennej lokalizacji siedziby i ciągłości jej istnienia. Brak dostępu do siedziby pociąga bowiem za sobą brak dostępu do docelowego komputera. Trwała zmiana lokalizacji komputera nadal wymaga utrzymywania siedziby, być może w odległym miejscu. Pewnym rozwiązaniem jest tu zastosowanie tradycyjnej usługi nazewniczej do lokalizacji siedziby. Ponieważ lokalizacja siedziby rzadko ulega zmianie, klienci mogą sobie tą informację przechowywać podręcznie, nie tracąc na wydajności.



Jednym z zastosowań mechanizmów lokalizacji jednostek jest telefonia mobilna (komórkowa). Podejście, które się tam stosuje polega na podejmowaniu próby lokalizacji jednostki z którą chcemy nawiązać łączność najpierw na podstawie lokalnego rejestru. Jeżeli to nie daje wyniku, nawiązywany jest kontakt z siedzibą jednostki w celu znalezienia jej aktualnego położenia. Schemat ten można uogólnić do wielopoziomowej hierarchii.

Rysunek przedstawia przykładową hierarchię sieci lokalizacyjnej. Sieć podzielona jest na domeny. Domeny najniższego poziomu (domeny-liście) odpowiadają lokalnej sieci komputerowej lub komórce w telefonii mobilnej. Każda domena ma swój węzeł katalogowy, gdzie przechowywana jest informacja o wszystkich jednostkach znajdujących się w całej domenie. Węzeł katalogowy korzenia zawiera informacje o wszystkich jednostkach w systemie. Każda jednostka reprezentowana jest przez rekord położenia. Rekord położenia w domenie-liściu zawiera aktualny adres jednostki. Rekord położenia w węzłach wyższego poziomu zawiera wskaźnik do węzła niższego poziomu, gdzie dana jednostka faktycznie się znajduje. W ten sposób budowany jest łańcuch wskazań od korzenia do domeny-liścia, w której jednostka się znajduje.



Jednostka może mieć wiele adresów. Informacja taka jest przechowywana w postaci wielu rekordów położenia znajdujących się w węźle katalogowym najmniejszej domeny zawierającej obie lokalizacje.

Najczęściej wykonywaną operacją w systemach nazewniczych jest oczywiście tłumaczenie nazwy na adres. W przypadku lokalizacji hierarchicznej zapytanie wędruje po drzewie hierarchii domen. Na rysunku zapytanie klienta zostało zgłoszone w zaznaczonej domenie na dole po lewej stronie. Ponieważ węzeł ten nie zawiera rekordu położenia dla poszukiwanej jednostki, następuje przekazanie żądania do węzła nadrzędnego (1). Ten również nie posiada odpowiedniego rekordu i przekazuje żądanie do swojego węzła nadrzędnego (2). Węzeł ten posiada już odpowiedni rekord położenia, ale jest on jedynie wskaźnikiem do węzła podrzędnego, następuje więc przejście do tego węzła (3). Kolejny węzeł zawiera również wskaźnik (4). W końcu zapytanie trafia do węzła zawierającego rekord położenia z adresem (4). Odpowiedź odsyłana jest bezpośrednio do klienta.

Poszukiwanie jednostek w modelu hierarchicznym wykorzystuje własność **lokalności**. Poszukiwanie obejmuje swoim zasięgiem coraz bardziej ogólne domeny, w skrajnym przypadku docierając do węzła katalogowego korzenia, który ma zarejestrowane wszystkie jednostki.

Wstawianie informacji o nowej jednostce (rysunek po prawej) przebiega podobnie do operacji wyszukiwania. Załóżmy, że żądanie wstawienia pojawiło się w tej samej domenie-liściu, co w poprzednim przypadku. Żądanie takie jest przekazywane do węzłów nadrzędnych, aż dotrze do węzła, który posiada już odpowiedni rekord położenia (kroki 1 i 2). Jeżeli jest to rejestracja pierwszej kopii jednostki, to żądanie takie może trafić aż do korzenia. W węźle, który zna już rejestrowaną jednostkę (lub w korzeniu) następuje dodanie nowego rekordu ze wskaźnikiem do węzła podrzędnego zawierającego jednostkę. Następnie, żądanie przechodzi tą samą ścieżkę do miejsca, z którego wyszło, dodając po drodze wskaźniki tworzące ścieżką prowadzącą do domeny-liścia (kroki 3 i 4). W domenie-liściu następuje dodanie odpowiedniego rekordu położenia z adresem nowej jednostki.

Usuwanie jednostki z usługi lokalizacyjnej przebiega podobnie do operacji wstawiania. Po usunięciu rekordu położenia w domenie-liściu następuje przechodzenie wzwyż hierarchii i usuwanie odpowiednich wskaźników. Jeżeli w danym węźle po usunięciu wskaźnika rekord położenia staje się pusty, to jest on również usuwany. Może nie być pusty, jeżeli w węźle znajdowało się kilka wskaźników, wskazujących na alternatywne lokalizacje. W takim przypadku operacja usuwania kończy się. Jeżeli jednostka była jedyną w systemie, następuje usunięcie wszystkich wskaźników i rekordów położenia aż do korzenia.

Przechowywanie podręczne w przypadku usług nazewniczych dla jednostek mobilnych może być nieefektywne, ponieważ dane te ulegają częstym zmianom, co powoduje ostatecznie nie odnajdywanie poszukiwanych jednostek. Przechowywanie podręczne daje jest korzystne jedynie wtedy, gdy dane rzadko się zmieniają.

W przypadku podejścia hierarchicznego może się okazać, że jednostka przemieszcza się, ale zawsze (czy prawie zawsze) w obrębie pojedynczej domeny. Jeżeli jest to domena-liść, to miejsce przechowywania adresu jednostki pozostaje cały czas to samo – jest to węzeł katalogowy tej domeny. Oznacza to, że ścieżka od korzenia do tego węzła jest stała i można ją przechować podręcznie. Zapytania o lokalizację tej jednostki mogą więc być kierowane bezpośrednio do węzła katalogowego, który zna odpowiedź. Jeżeli jednak przemieszczanie odbywa się na większym obszarze, to może następować zmiana przynależności do domen-liści. Nawet w tym jednak przypadku może istnieć domena wyższego poziomu, która obejmuje wszystkie odwiedzane przez jednostkę lokalizacje (i nie będzie to domena-korzeń). W takiej sytuacji można nadal stosować przechowywanie podręczne, ale dotyczące części ścieżki: od korzenia do domeny obejmującej wszystkie lokalizacje.

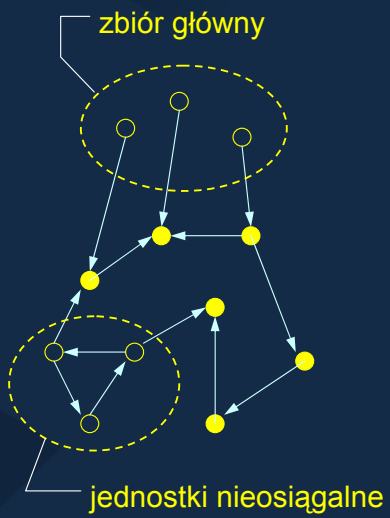
Na rysunku przedstawiono jednostkę, która znajduje się w domenie D2, ale przemieszcza się pomiędzy lokalizacjami w domenach D2 i D3. Domena D1 jest więc najmniejszą domeną obejmującą wszystkie możliwe lokalizacje jednostki. Adres węzła katalogowego D1 może więc być przechowywany podręcznie przez inne węzły w celu lokalizacji jednostki. Tak się dzieje w przypadku węzła katalogowego domeny D7. Dzięki przechowywaniu podręcznemu nie trzeba odwiedzać węzłów katalogowych domen D5 i D4 w celu lokalizacji wspomnianej jednostki.

Dodatkową optymalizacją możliwą do przeprowadzenia jest wymuszenie przechowywania docelowego adresu jednostki w węźle katalogowym domeny D1. Normalnie znajdowałby się tam jedynie wskaźnik do domeny D2 lub D3, w zależności od tego gdzie aktualnie znajduje się jednostka. Taka optymalizacja dodatkowo skraca czas pozyskania adresu.

Zastosowanie przechowywania podręcznego wskaźników może być pomocne, ale trzeba rozwiązać kilka problemów szczegółowych. Po pierwsze należy w jakiś sposób dokonać wyboru miejsca przechowywania adresu jednostki (w naszym przykładzie jest to węzeł katalogowy domeny D1). Po drugie należy dokonywać unieważniania wskaźników po rekonfiguracji danych w systemie nazewniczym. W powyższym przypadku rekonfiguracja byłaby potrzebna, gdyby powstała nowa kopia poszukiwanej jednostki np. w domenie D6. Zapytania zgłaszane w domenie D7 powinny wy tym przypadku być kierowane do nowej kopii, która jest najbliższa. Niestety nieaktualny wskaźnik przeniósłby poszukiwania do węzła domeny D1.

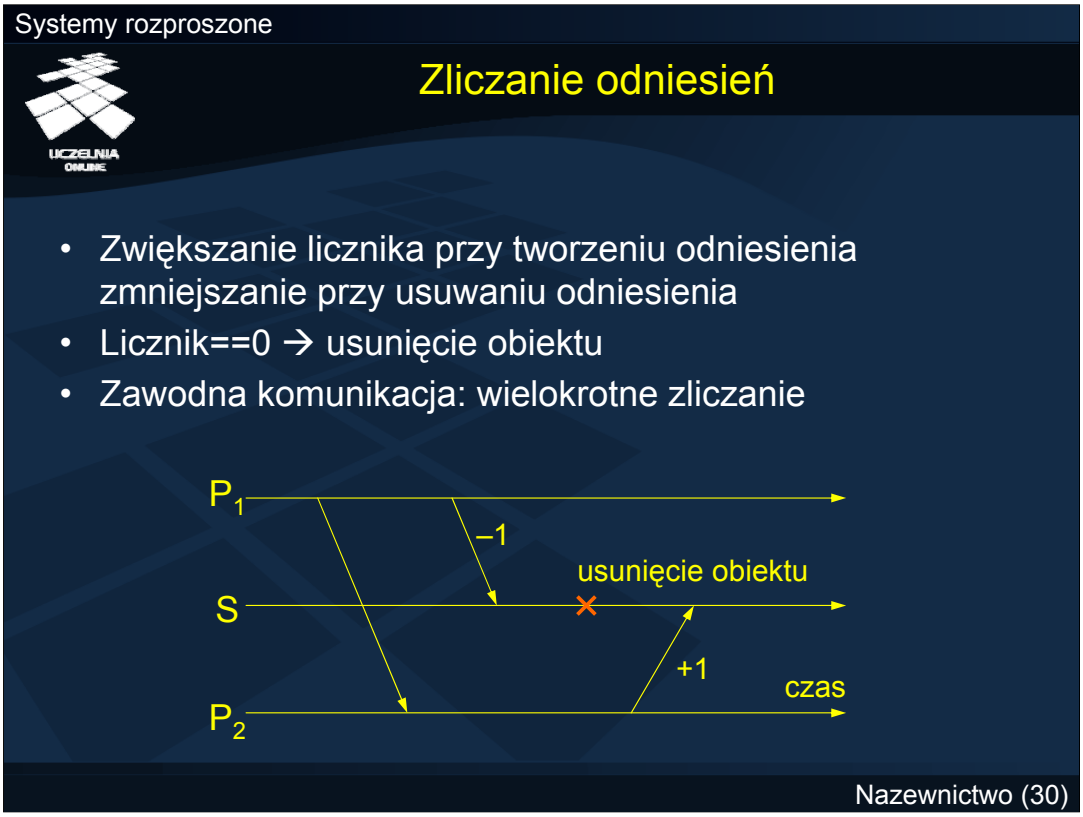
Usuwanie jednostek bez odniesień

- Odśmiecanie (ang. *garbage collection*)
- Obiekty bez odniesień
  - zbiór główny
  - obiekty nieosiągalne
- Rozproszone odśmiecanie
  - komunikacja (zawodna)
  - awarie



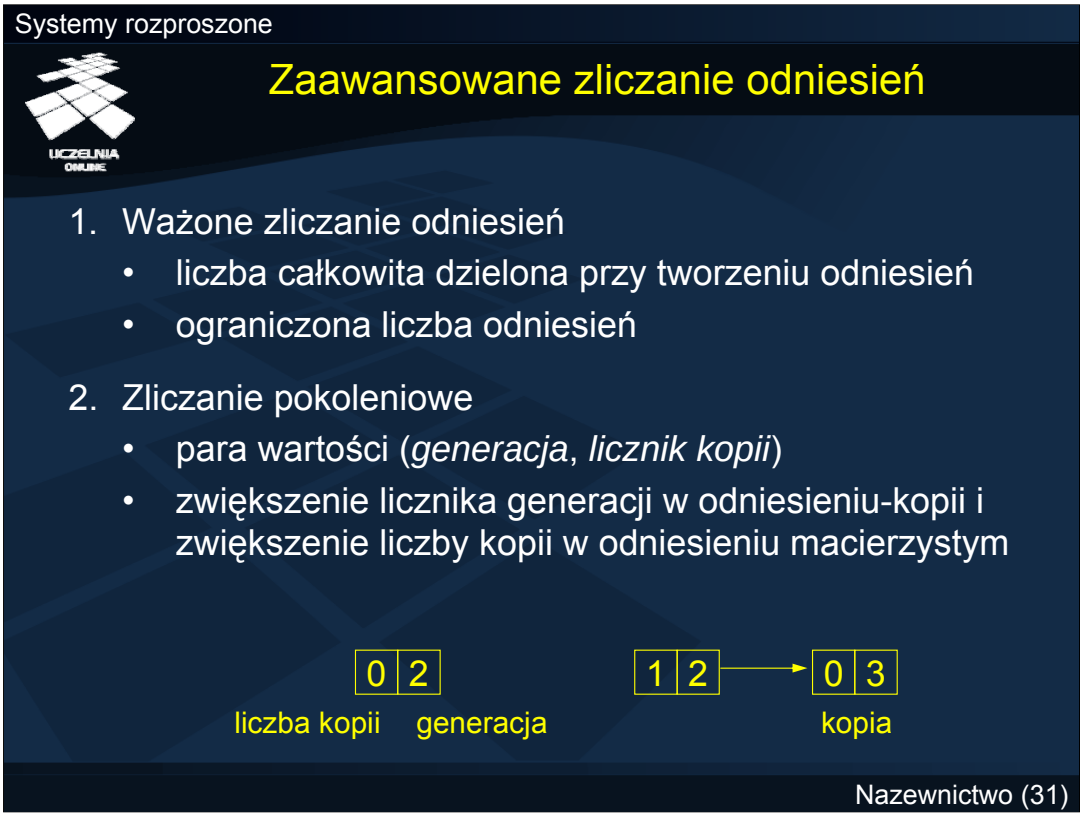
Usługi nazewnicze służą do lokalizacji jednostek. Jeżeli jednostka staje się niedostępna z powodu usunięcia wszystkich nazw odnoszących się do niej, powinna zostać usunięta. Jest to konieczne, aby zwolnić zasoby zajmowane przez tą jednostkę, zasoby które i tak nie są wykorzystywane. Działanie takie określa się mianem odśmiecania (ang. *garbage collection*), a w przypadku systemów rozproszonych – **rozproszonym odśmiecaniem**. Działanie to jest szczególnie istotne w systemach rozproszonych obiektów, gdzie nie jest realizowane automatyczne usuwanie obiektów, które nie są już używane. Fakt nieużywania obiektu można stwierdzić poprzez wykrycie braku odwołań do niego. W systemach rozproszonych oznacza to, że nie ma w systemie żadnego pośrednika (czyli obiektu emulującego obiekt po stronie klienta). Trzeba również uwzględnić sytuację, gdy dwa obiekty wskazują na siebie wzajemnie, ale do których nie ma żadnych odwołań w systemie. W przypadku ogólnym do usunięcia kwalifikują się wszystkie obiekty, które nie są osiągalne ze **zbioru obiektów głównych** (ang. *root set*). Można to zobrazować grafem powiązań pomiędzy obiektami, gdzie strzałka oznacza posiadanie odniesienia do obiektu wskazywanego.

W systemie scentralizowanym wykrywanie i usuwanie obiektów bez odniesień jest proste. W przypadku systemów rozproszonych operacja taka wymaga komunikacji, która może być wolna lub zawodna, co ogranicza w konsekwencji wydajność i/lub skalowalność poszczególnych rozwiązań.



W systemach scentralizowanych weryfikację wykorzystania obiektów można przeprowadzić stosunkowo prosto poprzez utrzymywanie licznika odwołań do obiektu. Każde utworzenie takiego odwołania powoduje zwiększenie licznika, a usunięcie odwołania jego zmniejszenie. Osiągnięcie wartości zero przez ten licznik oznacza, że obiekt jest niedostępny i można go usunąć. W systemie rozproszonym realizacja tej koncepcji może być problematyczna np. z powodu zawodnej komunikacji. Utrata komunikatu potwierdzającego zwiększenie bądź zmniejszenie licznika odwołań spowoduje ponowne wysłanie tego samego zlecenia, powielając w ten sposób operację na liczniku. Odpowiedni protokół musi więc zająć się usuwaniem podwójnych komunikatów.

Inny problem może wyniknąć z przekazywania odwołań pomiędzy procesami. Załóżmy, że proces P_1 przekazuje do procesu P_2 odwołanie do obiektu umieszczonego na serwerze S. Po wykonaniu tej operacji proces P_1 nie potrzebuje już obiektu, więc wysyła do serwera żądanie zmniejszenia licznika odwołań. Jeżeli było to jedyne odwołanie w systemie, to może nastąpić w tym momencie usunięcie obiektu (licznik osiągnie wartość 0), dlatego że proces P_2 nie zdążył jeszcze zarejestrować odwołania, które właśnie dostał od P_1. Rozwiązaniem tego problemu może być wprowadzenie dodatkowego komunikatu wysyłanego przez proces P_1 do serwera S *przed* przekazaniem odwołania do P_2, informującego, że będzie przekazywane odwołanie. Dzięki temu obiekt nie zostanie usunięty nawet, jeżeli jego licznik odwołań osiągnie wartość zero.



Problemu wyścigu (ang. *race condition*) występującego w zliczaniu odniesień można uniknąć poprzez wyeliminowanie komunikatów zwiększających licznik odwołań. Ważone zliczanie odniesień (ang. *weighted reference counting*) polega na ustaleniu początkowej liczby całkowitej dla każdego obiektu i dzielenie jej przy każdym tworzeniu nowego odniesienia. Przykładowo: jeżeli przypisano obiektowi wagę 128, to po skopiowaniu odniesienia oryginał i kopia będą miały wagę równą 64. Jeżeli odniesienie-kopia utworzy kolejną kopię to będą one miały wagi równe 32. Usuwanie odniesień powoduje przesłanie informacji o znikającej wadze. Jeżeli waga osiągnie wartość 0, oznacza to, że wszystkie odniesienia zostały usunięte i obiekt może być też usunięty.

Podstawową wadą tego podejścia jest ograniczona liczba kopii odniesień, które można utworzyć. Jeżeli waga osiąga wartość 1, to dalsze dzielenie nie będzie mogło być przeprowadzone. Rozwiązaniem jest wtedy wprowadzenie dodatkowego sztucznego odniesienia (wskaźnika naprowadzającego) do nowego obiektu-pośrednika, który będzie miał ustawioną wagę znów na dużą wartość. Usunięcie wszystkich odniesień do tego obiektu-pośrednika będzie powodować dopiero wysłanie komunikatu o usunięciu odniesienia do głównego obiektu. Podstawową wadą tej metody jest wydłużenie czasu dostępu przy długich łańcuchach wskaźników. Im dłuższy łańcuch tym większe jest również prawdopodobieństwo awarii.

Innym rozwiązaniem opartym na pośredniości jest **pokoleniowe zliczanie odniesień** (ang. *generation reference counting*). Każde odniesienie jest opatrzone podwójnym licznikiem, gdzie pierwsza wartość oznacza liczbę kopii utworzonych na podstawie tego odniesienia, a druga oznacza generację odniesienia. Jeżeli na podstawie odniesienia P tworzona jest kopia Q, to licznik kopii w odniesieniu P jest zwiększany, a numer generacji kopii Q jest inicjowany na wartość o jeden większą od licznika generacji w P. Usuwając odniesienie wysyłane są obie wartości liczbowe. Serwer obiektu przechowuje tablicę G, w której pozycja i-ta oznacza liczbę kopii i-tej generacji. Usuwając odniesienie z parą (i, k) zmniejszana jest o 1 wartość na pozycji G[i] i jednocześnie zwiększana o k na pozycji G[i+1]. Obiekt można usunąć gdy cała tablica G zostanie wyzerowana. Zaletą tego rozwiązania jest brak konieczności kontaktowania się z serwerem obiektu podczas tworzenia nowego odniesienia.

Wszystkie przedstawione tu rozwiązania zakładają niezawodną komunikację.

Spisywanie odniesień

- Jawny spis wszystkich odniesień
- Operacje dodawania i usuwania odnośników są idempotentne
- Stosowane w Java RMI
- Wyścig podczas przekazywania odniesienia
  - P_1 przekazuje odniesienie do P_2
  - P_1 usuwa odniesienie a P_2 dodaje
- Słaba skalowalność
  - ograniczenie czasowe – dzierżawa (ang. *lease*)

Zamiast zliczania odniesień można stosować jawne spisywanie wszystkich odniesień. Spis taki (ang. *reference list*) zachowuje własność *idempotentności*, która oznacza, że wielokrotne dodawanie czy usuwanie tych samych pozycji nie powoduje powstawania błędnych zapisów. W konsekwencji podejście to można zastosować w przypadku zawodnej komunikacji. Wymagane jest jedynie potwierdzenie wykonania operacji. Podejście to jest stosowane w Java RMI.

Spisywanie odniesień jest wrażliwe na problem wyścigu, podobnie jak to miało miejsce w zliczaniu odniesień. Jeżeli proces P_1 przekazuje odniesienie do P_2, to P_2 będzie rejestrował swoje nowe odniesienie na serwerze, a P_1 będzie wyrejestrowywał swoje. Jeżeli zgłoszenie procesu P_1 będzie pierwsze, to może się okazać, że lista odniesień stanie się na chwilę pusta, co możne spowodować usunięcie obiektu.

Podstawowa wada techniki spisywania odniesień polega na jej słabej skalowalności. Listy odniesień mogą się bowiem bardzo rozrastać. Aby to ograniczyć można stosować czasowe ograniczenie przechowywania informacji na liście, co jest określane jako **dzierżawa** (ang. *lease*). Odniesienie, które nie odnowi swojej dzierżawy jest automatycznie usuwane z listy.

Identyfikowanie jednostek nieosiągalnych

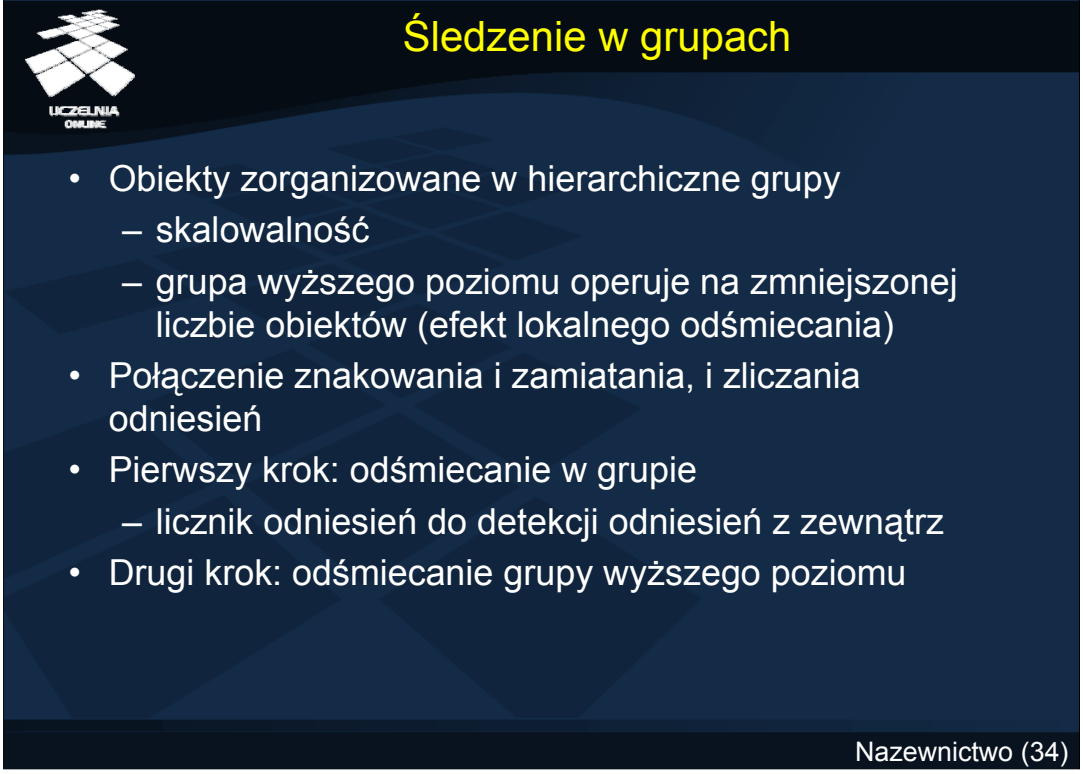
- Kolektor znakuj i zamiataj (ang. *mark-and-sweep*)
  - faza znakowania i faza usuwania
- Znakowanie trójkolorowe
  - biały (niedostępny), czarny (dostępny), szary (dostępny ale niesprawdzone odniesienia)
  - koniec odśmiecania: tylko biały i czarny
- Konieczność zatrzymania przetwarzania
  - odśmiecacze przyrostowe

W systemie rozproszonym może istnieć grupa obiektów, które wzajemnie na siebie wskazują, ale nie są osiągalne z zewnątrz. Takie obiekty również powinny zostać usunięte, ale metody przedstawione dotąd nie gwarantują ich usunięcia. Trzeba zastosować w tym przypadku **odśmiecanie ze śledzeniem** (ang. *tracing-based garbage collection*). Metody te są kosztowne i słabo skalowalne ponieważ wymagają śledzenia wszystkich jednostek w systemie.

W systemach scentralizowanych stosuje się kolektory typu **znakuj i zamiataj** (ang. *mark-and-sweep*). Kolektor działa w dwóch fazach. W pierwszej fazie wychodząc od obiektów ze zbioru głównego przechodzi rekurencyjnie wszystkie odniesienia znacząc obiekty, które odwiedza na czarno. W drugiej fazie obiekty, które nie zostały oznaczone są usuwane. W podejściu trójkolorowym kolor szary wykorzystuje się do zaznaczenia obiektów, które zostały już osiągnięte, ale dla których nie sprawdzono jeszcze wszystkich odniesień. Odwiedzenie wszystkich odniesień obiektu powoduje zmianę koloru na czarny.

Rozproszone znakowanie i zamiatanie odbywa się w sposób zrównoleglony. Odwiedzone obiekty po przetworzeniu lokalnym kolektorem stają się szare do czasu sprawdzenia wszystkich odwołań do zdalnych obiektów. Sprawdzanie odwołań do zdalnych obiektów powoduje wysłanie komunikatu do zdalnego obiektu informującego o osiągalności. Po przetworzeniu wszystkich odniesień w obiekcie zdalnym odsyłane jest potwierdzenie. Przetworzenie wszystkich odniesień do obiektów zdalnych powoduje zmianę koloru na czarny.

Podstawową wadą znakowania i zamiatania jest konieczność całkowitego zablokowania przetwarzania na czas odśmiecania. W systemie rozproszonym oznacza to konieczność synchronizacji wszystkich rozproszonych procesów, co może być nieakceptowalne dla użytkowników. Rozwiązaniem może być zastosowanie odśmiecaczy przyrostowych. Niestety nie są one dobrze skalowalne. Działają one współbieżnie z aplikacjami, modyfikującymi graf osiągalności, co powoduje częste znakowanie obiektów kolorem szarym i komunikację między węzłami, która jednak nie prowadzi do efektywnego wykrywania nieosiągalnych obiektów.



Mechanizm rozproszonego odśmiecania można zoptymalizować poprzez zorganizowanie przestrzeni obiektów w hierarchiczne grupy. Odśmiecanie jest wtedy realizowane najpierw lokalnie w każdym procesie, następnie w lokalnej grupie, a później w grupach wyższego poziomu. Do analizy odniesień wykorzystywane są liczniki odwołań; następuje porównanie liczby odniesień do obiektu z liczbą obiektów-pośredników odnoszących się do tego obiektu w danej grupie. W ten sposób można stwierdzić fakt istnienia odniesień spoza grupy, a więc odniesień, które będą musiały być przeanalizowane w kolejnym kroku. Lokalne odśmiecanie pozwala jednak wyeliminować z analizy sporą grupę obiektów, co pozwala na zredukowanie liczby węzłów w grafie analizowanym na wyższym poziomie. Jest to podstawowa przesłanka do stosowania hierarchicznych grup. W efekcie uzyskuje się wyższą skalowalność algorytmu – odśmiecanie realizowane jest bowiem na podobnej liczbie obiektów pomimo przechodzenia do wyższych grup w hierarchii.