

15. OCHRONA ZASOBÓW

Zagadnienia:

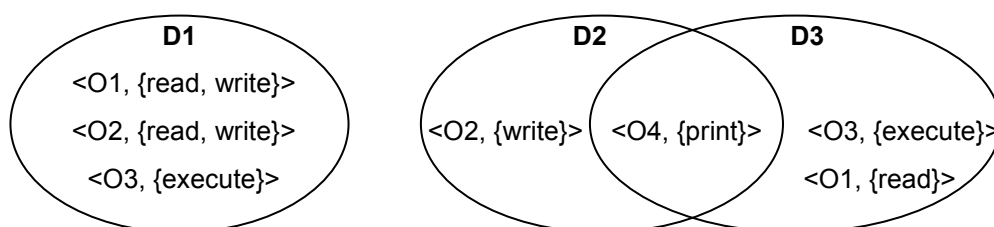
- Istota i cele ochrony zasobów.
- Koncepcja domen.
- Macierz dostępu.
- Implementacja macierzy dostępu.
- Egzekwowanie praw dostępu.

1. Istota i cele ochrony zasobów.

System operacyjny, zarówno lokalny jak i rozproszony, składa się z bardzo wielu obiektów (sprzętowych i programowych) stanowiących zasoby różnych klas. Każdy z tych zasobów, oprócz posiadania jednoznacznej „tożsamości” (ma własną nazwę), może być używany (udostępniany) za pomocą ściśle zdefiniowanego repertuaru operacji. Istota ochrony zasobów polega na tym, aby zapewnić ich używanie w sposób zgodny z przeznaczeniem i poprawny oraz umożliwić to tylko tym procesom (procesem jest też sesja użytkownika), które mają odpowiednie uprawnienia. Rozwiązanie tego problemu jest szczególnie trudne w rozproszonych systemach operacyjnych, aczkolwiek stanowiło poważne wyzwanie już od momentu zaistnienia pierwszych systemów obsługujących wielu użytkowników (UNIX jest bardzo typowym przykładem, miał jednak poprzedników).

2. Koncepcja domen.

Przez domenę rozumiemy (w kontekście ochrony zasobów) zbiór praw dostępu do poszczególnych obiektów (zasobów systemu). Prawa dostępu definiowane są jako relacja typu: $\langle \text{nazwa_obiektu}, \{\text{zestaw_uprawnień}\} \rangle$. Domeny mogą być między sobą we wzajemnych relacjach (zobacz Rys. 15.1.). W praktyce domeny odpowiadają użytkownikom lub grupom użytkowników.



Rys. 15.1.

Przykłady domen i relacji między domenami (O1, ..., O4 – obiekty; D1, ..., D3 – domeny).

W przypadku systemów klasy UNIX implementacja takiego systemu wygląda następująco:

- System składa się z dwóch typów domen: zwykły użytkownik (*user*) i administrator (*supervisor, root*).
- Nazwę domeny w UNIX-ie określa identyfikator użytkownika (*uid*).
- Przełączanie domen w trakcie działania procesu odbywa się za pomocą systemu plikowego:
 - Każdy plik (oprócz innych atrybutów) posiada znacznik domeny (bit *setuid*), który jest istotny dla plików wykonywalnych.

- Jeśli taki plik wykonywalny programu jest ładowany (powstaje „z niego” proces) a jego znacznik `setuid` jest ustawiony na 0 (normalna sytuacja), to identyfikator `uid` wywołującego program użytkownika jest ustawiany jako identyfikator właściciela procesu (identyfikator właściciela pliku w trakcie wykonywania), aż do zakończenia procesu. Dzięki temu np. zwykły użytkownik może zmienić hasło za pomocą programu `passwd` tylko sobie.
- Jeśli bit `setuid` pliku wykonywalnego jest ustawiony na 1, to właścicielem procesu pozostaje właściciel pliku (np. użytkownik `root`). Uprawnienia procesu do zasobów systemu operacyjnego (np. plików) nie są więc takie, jak użytkownika który uruchomił proces, lecz takie, jak właściciela pliku wykonywalnego – np. użytkownika `root` gdyby dotyczyło to programu `passwd`.

3. Macierz dostępu.

Uprawnienia do konkretnych obiektów, dostępne w ramach konkretnych domen można przechowywać w strukturze dwuwymiarowej tablicy (macierzy uprawnień – ang. *access matrix*). W tej macierzy odpowiednio:

- wiersze reprezentują domeny,
- kolumny reprezentują obiekty.

Element macierzy uprawnień ($Access[i, j]$) o indeksach i, j (zobacz Rys. 15.2.) reprezentuje więc zestaw operacji, które proces wykonujący się w domenie i ma prawo wykonać na obiekcie j .

Obiekt	O ₁ (Plik ₁)	O ₂ (Plik ₂)	O ₃ (Plik ₃)	O ₄ (Drukarka)
Domena				
D ₁	odczyt, zapis	odczyt, zapis	wykonanie	
D ₂		zapis		drukowanie
D ₃	odczyt		wykonanie	drukowanie
D ₄	odczyt, zapis	odczyt, wykonanie	odczyt, zapis	

Rys. 15.2.

Przykład macierzy dostępu.

Działanie macierzy dostępu polega na tym, że jeśli domena D_i stara się wykonać jakąś operację (OP) na obiekcie O_j , to system pozwoli na to pod warunkiem, że operacja OP znajduje się na odpowiedniej pozycji macierzy.

Macierz dostępu może ulegać zmianom zarówno na skutek działań administratora systemu, jak i innych użytkowników. Typowe funkcje zmieniające stan macierzy dostępu to:

- przydzielenie i usunięcie praw dostępu (operacje podstawowe),
- szczególne operacje na zawartości macierzy dostępu:
 - relacja własności: domena D_i jest właścicielem obiektu O_j ,
 - skopiowanie dopuszczalnej operacji OP z obiektu O_j do O_k ,
 - zarządzanie domeną: D_i (np. `root`) może modyfikować uprawnienia D_i ,
 - przełączanie: przejście z domeny D_i do domeny D_m (np. zmiana grupy).

Można powiedzieć, że system zarządzania dostępem do zasobów oparty na macierzy dostępu składa się z dwóch warstw:

- zbioru abstrakcyjnych reguł (nazywanych np. polisą użytkownika), określających kto może co zrobić z którym obiektem (reguły te związane są najczęściej z użytkownikami lub ich grupami),
- mechanizmu wykonawczego, czyli zaimplementowanej (nie koniecznie dosłownie) w systemie operacyjnym macierzy dostępu, zawierającej „mapę” uprawnień oraz odpowiednich narzędzi, zapewniających bezpieczne (tj. możliwe do wykonania tylko przez uprawnionych użytkowników) ustawianie tych uprawnień i ich ścisłe przestrzeganie.

4. Implementacja macierzy dostępu.

Każda kolumna w macierzy dostępu stanowi listę dostępu do danego obiektu (ang. *access list*), definiuje kto ma prawo wykonać na tym obiekcie jakie operacje.

Każdy wiersz w macierzy dostępu stanowi listę uprawnień danego użytkownika (ang. *capability list*), definiuje co może on zrobić z którym obiektem.

Zarządzanie uprawnieniami użytkowników polega przede wszystkim na wykonywaniu na macierzy dostępu podstawowych operacji umieszczania lub usuwania uprawnień do konkretnych operacji (np. odczyt, zapis, wykonywanie) w odpowiednich komórkach tablicy. W podobny sposób można realizować własność obiektu – wystarczy ustawić znacznik atrybutu własności w „na wysokości” wybranej domeny. Zauważmy przy tym, że w systemach klasy UNIX plik jest jednocześnie własnością jednego użytkownika i jednej grupy, ewentualnie może nie mieć właścicieli.

Funkcją, która może ułatwić pracę administratora systemu jest kopiowanie całych wierszy lub ich wybranych fragmentów, co pozwala łatwo nadawać te same lub zbliżone uprawnienia różnym użytkownikom. Kopiując całe kolumny możemy z kolei szybko powielać zasady dostępu do różnych obiektów (np. plików).

Realizacja operacji zarządzania domenami (czyli w praktyce użytkownikami i grupami) oraz przełączania między domenami możliwa jest dzięki dołączeniu domen do obiektów stanowiących zasoby systemu. Uzyskujemy to dodając do macierzy kolumny reprezentujące domeny (zobacz Rys. 15.3.).

Obiekt	O ₁ (Plik ₁)	O ₂ (Plik ₂)	O ₃ (Plik ₃)	O ₄ (Drukarka)	D ₁	D ₂	D ₃	D ₄
Domena								
D ₁	odczyt, zapis	odczyt, zapis	wykonanie			przełącz		
D ₂		zapis		drukowanie			przełącz	przełącz, zarządzaj
D ₃	odczyt		wykonanie	drukowanie				
D ₄	odczyt, zapis	odczyt, wykonanie	odczyt, zapis		przełącz			

Rys. 15.3.

Przykład zmodyfikowanej macierzy dostępu z domenami w charakterze obiektów.

5. Egzekwowanie praw dostępu.

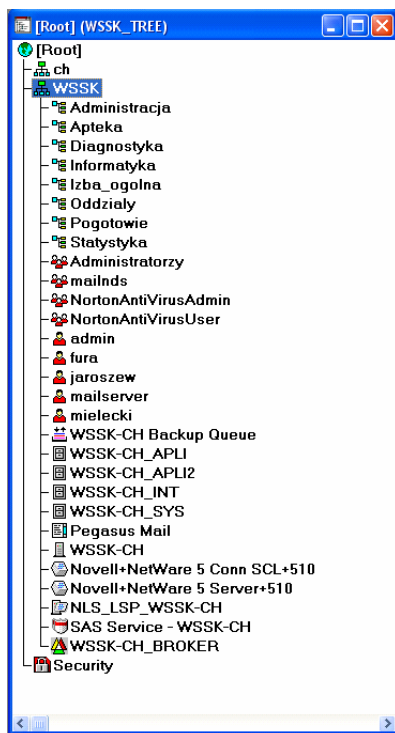
Warto zauważyć, że implementacja macierzy dostępu nie koniecznie musi polegać na zdefiniowaniu i obsługiwaniu odpowiedniej tablicy (problemem byłyby np. zmieniające się wymiary takiej tablicy, nie mówiąc już o jej wielkości i dużej liczbie pustych komórek). W systemach klasy UNIX ochrona dostępu do plików zrealizowana jest w ścisłym powiązaniu z samym systemem plikowym – atrybuty własności oraz uprawnień dostępu zapisywane są w strukturach opisujących pliki (i-węzłach). Dzięki temu bardzo łatwo jest zbudować listę dostępu do danego pliku (kolumnę macierzy uprawnień opisującą dany plik) w momencie żądania dostępu do niego. Egzekwowanie praw dostępu do plików realizowane jest tu więc na podstawie list dostępu, nie list uprawnień użytkownika.

Zauważmy, że odwrotne podejście zmuszałoby nas do utrzymywania dużych tablic pamiętających listy uprawnień (teoretycznie dla każdego użytkownika tyle elementów, co plików). Kłopotliwe byłoby zarówno pamiętanie takich list jak i ich obsługa.

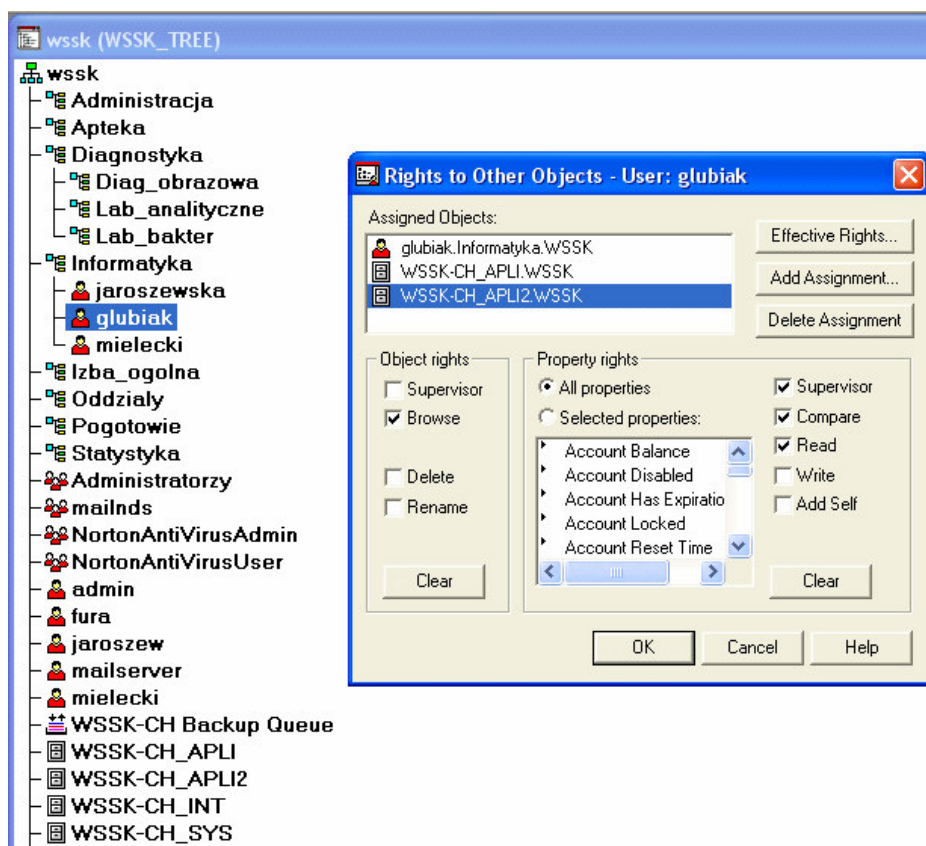
Przedstawiona koncepcja nie jest ani jedynym, ani najbardziej zaawansowanym technologicznie sposobem sterowania dostępem do zasobów systemu. We współczesnych, zwłaszcza sieciowych oraz rozproszonych systemach operacyjnych zastosowanie znalazły obiektowe bazy danych (z wykorzystaniem m.in. dziedziczenia uprawnień). Standardem stało się również katalogowanie wszystkich zasobów systemu według ujednoliconej zasady (np. zgodnie z założeniami protokołu LDAP – ang. *Lightweight Directory Access Protocol*). W tak skonstruowanych systemach zarządzania uprawnieniami mamy do czynienia nie tyle z listami dostępu czy listami uprawnień, co ze ścieżkami wytyczonymi w drzewiastej strukturze bazy danych. Drzewo jest strukturą danych znacznie bardziej elastyczną od tablicy czy też listy, znane są przy tym efektywne algorytmy obsługi drzew. W efekcie istnieje możliwość względnie szybkiego zbudowania listy uprawnień w kontekście danego użytkownika (z uwzględnieniem uprawnień dziedziczonych przez niego od obiektów nadrzędnych – grup, jednostek organizacyjnych itp.).

Przykładem takiego podejścia jest system katalogowania zasobów za pomocą rozproszonej, obiektowej bazy danych, zaproponowany przez firmę Novell i określany jako NDS (ang. *Netware Directory Services*), obecnie (po uzgodnieniu z LDAP) znany pod handlową nazwą *eDirectory*. Zasoby sieciowego systemu operacyjnego skatalogowane są tu w strukturze drzewa (zobacz Rys. 15.4.). W celu zorganizowania zasobów w sposób odzwierciedlający organizację przedsiębiorstwa lub instytucji wprowadzono dwie abstrakcyjne klasy obiektów: „organizacja” i „jednostka organizacyjna”. Klasy te nie reprezentują konkretnych zasobów sprzętowych, są jedynie kontenerami, w których można umieszczać różne zasoby (serwery, wolumeny dyskowe, użytkowników, grupy użytkowników i wiele innych). Przyjęto przy tym, że kontener klasy „organizacja” (ang. *Organization* – O) nie może zawierać innych kontenerów tej klasy, jedynie kontenery klasy „jednostka organizacyjna” (ang. *Organizational Unit* – OU) lub obiekty – „liście drzewa”, czyli np. użytkowników, grupy, serwery, wolumeny dyskowe, drukarki sieciowe i in.). Za to kontener klasy „jednostka organizacyjna” może zawierać dowolną liczbę następnich jednostek organizacyjnych oraz oczywiście obiekty typu „liście”. Gwoli ścisłości należy dodać, że istnieje tu jeszcze jedna klasa abstrakcyjna, będąca kontenerem najwyższego poziomu – „państwo” (ang. *Country* – C). Administratorzy raczej nie robią z niej jednak użytku.

Umiejscowienie konkretnego (dowolnego) zasobu w takim katalogu wymaga podania tzw. kontekstu, czyli ścieżki wytyczonej w drzewie i zawierającej nazwy kolejnych kontenerów. Np. użytkownik „glubiak” należy do kontekstu: O = WSSK, OU = Informatyka (zobacz Rys. 15.5.), co zapisujemy w postaci ścieżki `glubiak.Informatyka.WSSK` (jest to przykład bardzo prosty).

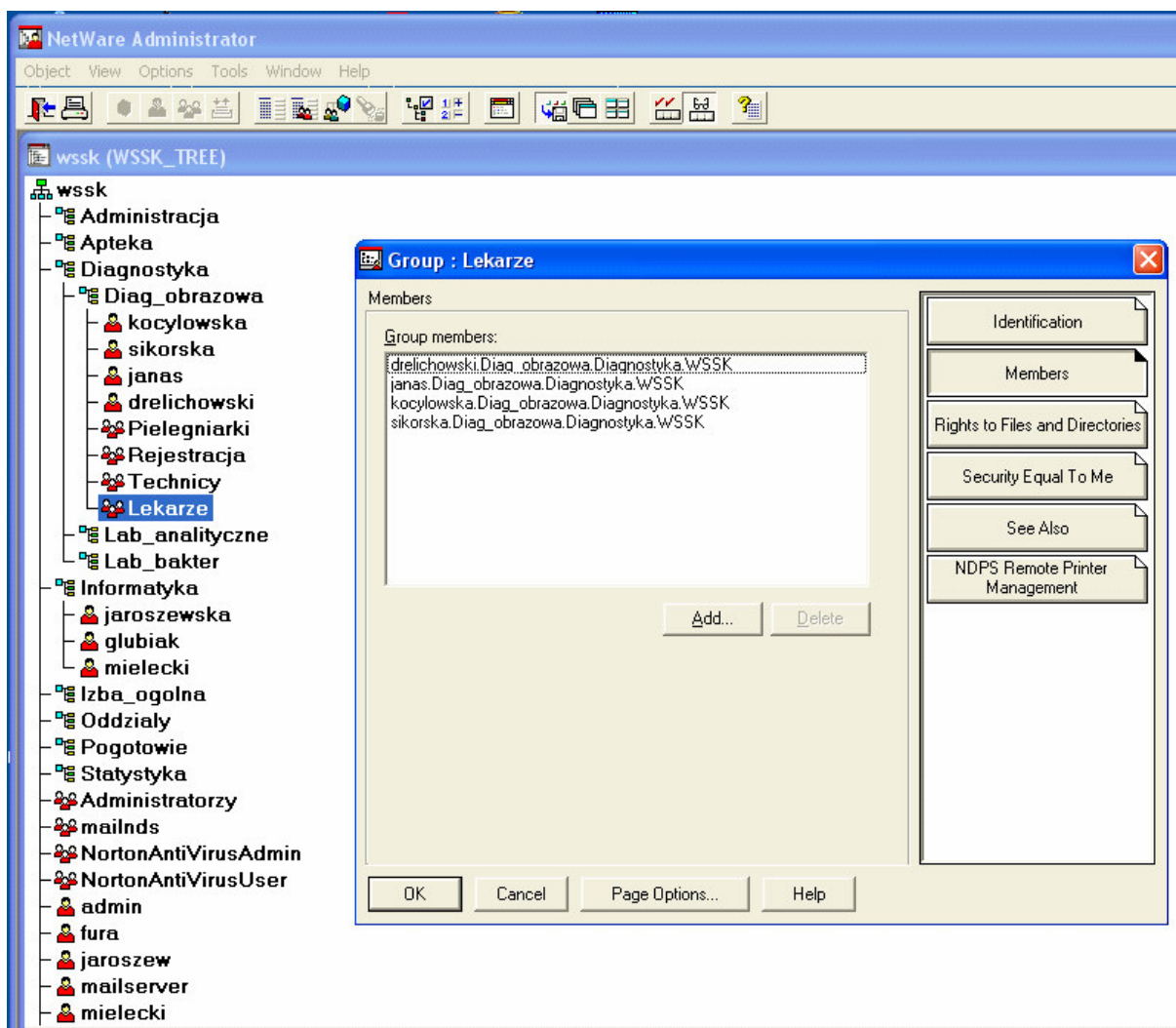


Rys. 15.4.
Prosty przykład drzewa NDS w systemie NetWare (obiekty klasy OU nie są rozwinięte).



Rys. 15.5.
Przykład definiowania uprawnień do wybranych obiektów dla użytkownika z kontekstu OU=Informatyka, O=WSSK w drzewie WSSK_TREE.

Zarządzanie uprawnieniami użytkowników w takim systemie polega (pozornie!) na ułożeniu listy dostępu dla użytkownika, a więc odwrotnie niż w przypadku systemów klasy UNIX. W rzeczywistości mamy tu do czynienia nie ze statyczną (i bardzo długą) tablicą, lecz z dynamiczną listą generowaną na podstawie wpisów do obiektowej bazy danych o strukturze drzewa. W praktyce administratorzy dodatkowo upraszczają sobie pracę definiując w każdym kontekście najpierw grupy użytkowników, następnie przypisują tym grupom uprawnienia do zasobów. Na końcu zakłada się konta użytkownikom w poszczególnych kontekstach, zamiast jednak tworzyć indywidualnie listy dostępu dla każdego z nich dopisuje się ich po prostu do odpowiednich, czasem kilku jednocześnie, grup (zobacz Rys. 15.6).

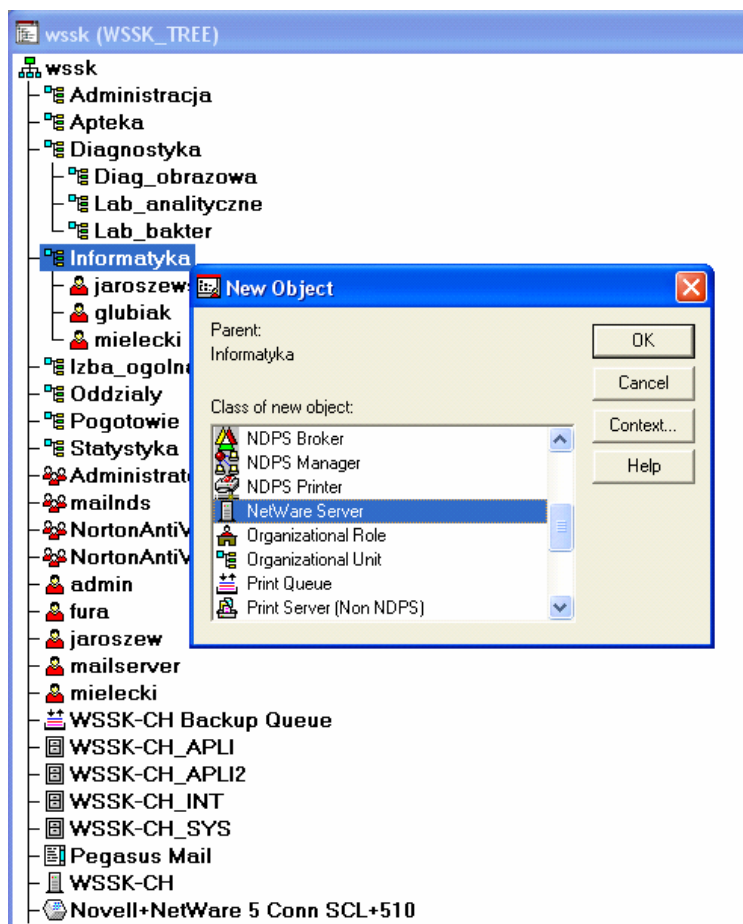


Rys. 15.6.

Panel zarządzania atrybutami grupy użytkowników w aplikacji „NetWare Administrator”.

Należy zauważyć, że przedstawiona powyżej technika zarządzania zasobami rozproszonego systemu stawia przed administratorem wysokie wymagania. Projekt drzewa musi zostać starannie przemyślany, z uwzględnieniem m.in. takich aspektów, jak topologia używanej sieci, umiejscowienie zasobów w stosunku do potencjalnie najczęściej je wykorzystujących użytkowników, zachowanie się systemu w sytuacjach awaryjnych (np. brak dostępu do któregoś z serwerów), możliwość rozbudowy o nowe jednostki organizacyjne i zasoby (skalowalność) i in.

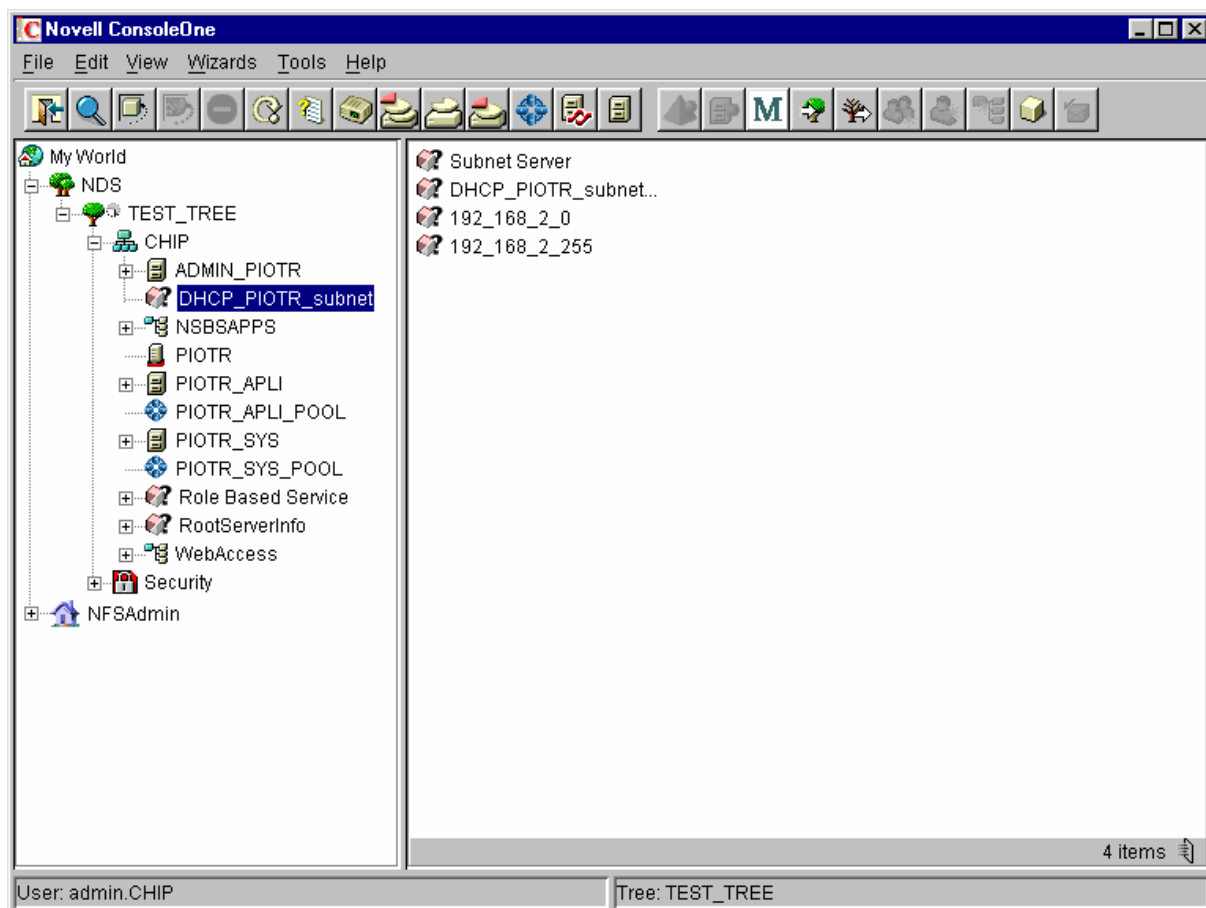
Istotną zmianą, wynikającą z rosnącego obciążenia, jest np. dodanie nowego serwera przeznaczonego do obsługi katalogów dyskowych i drukarek używanych przez określoną jednostkę organizacyjną. Z reguły optymalnym rozwiązaniem jest „logiczne” umieszczenie nowego serwera nie bezpośrednio przy korzeniu drzewa, lecz w kontekście danej jednostki organizacyjnej (zobacz Rys. 15.7.). Jeżeli jeszcze zadbamy o fizyczne umieszczenie serwera np. w budynku, w którym ma swoją siedzibę jednostka, to istnieje szansa, że jej pracownicy będą mogli pracować nawet w przypadku poważnej awarii sieci.



Rys. 15.7.

Umieszczenie nowego serwera w kontekście jednostki organizacyjnej.

Zainstalowanie oraz administrowanie takim systemem, także nanoszenie mniej lub bardziej istotnych modyfikacji na strukturę drzewa, byłoby praktycznie niemożliwe bez odpowiednich narzędzi administratora. Aczkolwiek możliwe jest np. „ręczne” zainstalowanie a następnie skonfigurowanie protokołu LDAP i zarządzanie w ten sposób dużą siecią z serwerami klasy UNIX / Linux (oraz Sambą jako serwerem zasobów plikowych i drukarkowych), to nakład pracy z tym związany praktycznie uniemożliwia skuteczne administrowanie dużymi sieciami. W przypadku systemów z rodziny Windows NT / 2000 itp. uciążliwa jest z kolei konieczność posługiwania się rozbudowanym zestawem narzędzi, częściowo pokrywających się funkcjami. Firma Novell, aczkolwiek od szeregu lat wyraźnie tracąca rynek, oferuje jednak wciąż najlepsze narzędzia administratora, integrujące bardzo szeroki zakres funkcji związanych z zarządzaniem siecią (np. protokołem DHCP), oraz zasobami (skatalogowanymi za pomocą LDAP). Najważniejszym obecnie narzędziem administratora jest napisana w Javie aplikacja ConsoleOne (zobacz Rys. 15.8.).



Rys. 15.8.
Zintegrowany pulpit administratora systemu NetWare – aplikacja ConsoleOne.