ZESZYTY NAUKOWE POLITECHNIKI ŚLĄSKIEJ 1997
Seria: INFORMATYKA z.32 Nr kol. 1356

Piotr KLUCZWAJD, Jarosław ULCZOK
Politechnika Śląska, Instytut Informatyki

NADZOROWANIE I REJESTRACJA OBCIĄŻENIA SIECI W SYSTEMACH OPERACYJNYCH WINDOWS 95 I WINDOWS NT

Streszczenie. Niniejszy artykuł jest próbą analizy możliwości monitorowania i kontrolowania pracy sieci komputerowych, w których ruch jest generowany w dużej mierze przez komputery pracujące pod kontrolą 32-bitowych systemów operacyjnych Windows 95 oraz NT z wykorzystaniem biblioteki zgodnej ze specyfikacją Winsock w wersji 1.1 i wyższych. Dołączony został również opis zrealizowanego i przetestowanego modelu monitora.

ANALYSING AND MONITORING OF NETWORK TRAFFIC IN WINDOWS 95 AND WINDOWS NT OPERATING SYSTEMS

Summary. This article is a trial of analysis of monitoring and control possibilities in networks, where data flow is generated in most by computers working under control of 32-bit operating systems Windows 95 and NT using a WINSOCK library compliant with 1.1 or higher version of specification. There is also included a description of a prepared and tested monitor model.

1. Wstęp

Jednym z największych sukcesów techniki czasów nam współczesnych, a zarazem symbolem postępu i nowoczesności w dziedzinie łączności i integracji stała się ogólnoświatowa sieć Internet. Dając ogromne możliwości wymiany danych, nie ma sobie równych. Jednak korzystanie z udogodnień, serwisów informacyjnych oraz ogromnych zasobów serwerów plików powoduje wzmożony przepływ danych przez łącza, których ograniczona przepustowość w dużym stopniu zmniejsza szybkość transmisji i rzutuje na komfort pracy.

W sytuacji kiedy apetyty użytkowników bezustannie wzrastają, przy znacząco wolniejszym tempie postępu technicznego oraz ograniczonych możliwościach rozbudowy i modernizacji istniejących struktur, należałoby zastanowić się, w jaki sposób można nadzorować i ewentualnie redukować ruch w sieci oraz kontrolować poprawność pracy sieci komputerowej przez rejestrowanie i sygnalizowanie w czasie rzeczywistym występujących błędów.

W chwili obecnej jedną z najbardziej popularnych rodzin 32-bitowych systemów operacyjnych jest rodzina Windows (95 i NT). Prawdopodobne zatem jest, iż część ruchu generowanego w sieciach komputerowych jest wynikiem funkcjonowania tych właśnie systemów. Zatem proces kontrolowania i monitorowania pracy części sieci mógłby odbywać się w tych środowiskach.

Dostępne standardowo w systemie Windows oprogramowanie pozwala na ograniczone kontrolowanie pracy sieci. Z systemem otrzymujemy kilka prostych narzędzi:

- **arp** - wyświetla i pozwala modyfikować tablicę konwersji adresów IP na Ethernet lub Token Ring używaną przez protokół ARP,
- **ipconfig** - wyświetla wszystkie aktualne ustawienia dotyczące TCP/IP dla danego komputera; można sprawdzić maskę podsieci, adres IP, adres domyślnego gatewaya, itp.,
- **nbtstat** - podaje połączenia wykonane przy pomocy protokołu NetBIOS symulowanego przy pomocy TCP/IP,
- **netstat** - pozwala na obejrzenie listy aktualnych połączeń wykonanych przy pomocy TCP/IP (połączenia z serwerem przeważnie nie są pokazywane) oraz całej statystyki Ethernet-u (ilość ramek, błędów, przesłanych bajtów, itp.).

Powszechnie dostępne oprogramowanie pozwala z reguły na powierzchowne kontrolowanie pracy sieci. Nie można gromadzić informacji przez zadany okres i z zadaną częstotliwością (tu wyjątkiem jest netstat). Jeżeli dodatkowo porównamy to z czasochłonnością ewentualnych zestawień, to okaże się, że nadzorowanie pracy sieci może być bardzo uciążliwe - co dyskwalifikuje wymienione narzędzia przy wykorzystywaniu ich do stałego monitorowania.

Analizując rynek produktów komercyjnych wykorzystywanych do kontroli pracy sieci komputerowych, należy stwierdzić, że dotychczas przy ich tworzeniu główny nacisk został położony na administrowanie urządzeniami z wykorzystaniem protokołu SNMP, zaś wdrożeń pozwalających na kontrolowanie obciążenia sieci wykorzystujących, np. protokół RMON+, jest wciąż niewiele. Ponadto dostępne produkty posiadają rozbudowane cechy funkcjonalne, np. Sniffer firmy Network General oferuje statystyki dotyczące wszystkich warstw siedmiowarstwowego modelu odniesienia ISO/OSI oraz rozpoznaje ponad 250 protokołów

sieciowych, za które to cechy, nawet jeśli ich nikt nie wykorzysta, trzeba zapłacić niebagatelne kwoty, a co istotne - do nielicznych należą rozwiązania pracujące w środowisku Windows 95.

Pokrywające duży obszar zastosowań byłoby oprogramowanie śledzące pracę sieci TCP/IP, w której komputery pracują pod kontrolą 32-bitowego systemu Windows. Protokoły te ze względu na ogromną popularność i możliwość zastosowania w sieciach rozległych zaczynają powoli wypierać lub uzupełniać inne rozwiązania. Nadzór pracy sieci należałoby rozproszyć i prowadzić indywidualnie na każdym komputerze, tak aby nie powodować dodatkowego obciążenia sieci komputerowej i w minimalnym stopniu absorbować część czasu i zasobów systemu. Ponadto informacje gromadzone w określonym okresie czasu w poszczególnych jednostkach pozwalałyby na dokładniejsze analizy występujących usterek w odróżnieniu od monitora centralnego - np. jednego dla całej sieci instalowanego na routerze, zaś jak najbogatsze statystyki mogłyby być punktem wyjściowym analiz pracy sieci, stawiania diagnoz czy też podejmowania decyzji z nią związanych.

1.1. Możliwości zastosowania monitora sieci

- Badanie obciążenia sieci - śledzenie adresów początkowych i końcowych,
- Kontrola adresów docelowych,
- Blokowanie niepożądanej komunikacji (konkretnych adresów, protokołów, itp.),
- Rozliczanie odpłatnych usług udostępnianych pod określonymi adresami sieciowymi,
- Rozliczanie ruchu w sieci,
- Kontrola zdarzeń powodujących nadmierne obciążenie sieci (najdłużej trwająca transmisja, najwięcej przesłanych danych, najwięcej otwartych połączeń).

Obecnie ze względu na ogromne obciążenie różnego rodzaju sieci w wyniku zbyt małej przepustowości w stosunku do potrzeb i oczekiwań użytkowników najbardziej celowa wydaje się być kontrola obciążenia oraz prowadzenie na jej podstawie różnorodnych statystyk pozwalających administratorom na podejmowanie decyzji z zakresu:

- okresowego wstrzymywania ruchu w czasie najmniejszych obciążeń np. w celu przeprowadzenia konserwacji sprzętu,
- rozplanowania środków na nowe inwestycje - można podjąć decyzję czy potrzebny jest nam bardziej wydajny serwer czy też zwiększenie przepustowości łączy,
- ewentualnych obostrzeń w dostępie do sieci - np. dla użytkowników, powodujących nieuzasadnione przeciążanie sieci.

2. Metody rejestrowania komunikacji z wykorzystaniem protokołu TCP/IP w środowisku Windows NT i Windows 95

Proces rejestrowania ruchu w sieci generowanego przez urządzenie należałoby prowadzić z wykorzystaniem śledzenia wywołań funkcji API (*ang. Aplication Programming Interface*) biblioteki WSOCK32.DLL w wersji 1.1, aktualnie najszerzej wykorzystywanej, odpowiedzialnej za obsługę warstwy transportowej TCP/IP. Kontrola ta może zostać rozszerzona na inne protokoły w wypadku wdrożenia wersji 2.0 tej specyfikacji, która nie ogranicza się jedynie do protokołów TCP/IP, ale również do NetBIOS, IPX, SPX, itp. Dzięki temu będzie możliwe za pomocą tej metody kontrolowanie pracy sieci wykorzystującej np. kilka różnych protokołów

Dostępne obecnie oprogramowanie śledzące pracę biblioteki WINSOCK to proste debuggery pozwalające na rejestrację: wywołań funkcji API, czasu trwania poszczególnych wywołań, wartości parametrów aktualnych wywołań oraz zwracanych rezultatów.

Ponadto większość producentów oprogramowania wykorzystującego bibliotekę WINSOCK daje możliwość śledzenia (ale tylko w wypadku komunikowania się z wykorzystaniem swojej implementacji biblioteki) m.in. takich parametrów jak:

- liczba wysłanych i odebranych pakietów protokołów: ARP (Adres Resolution Protocol), IP (Internet Protocol), ICMP (Internet Control Message Protocol), UDP (User Datagram Protocol), TCP (Transmission Control Protocol), SNMP (Simple Network Management Protocol)
- zawartość tablic: ARP, Routingu, Gniazdek (wykaz aktualnie wykorzystywanych), SNMP, DNS.

Na podstawie takich rejestrów trudno jest postawić diagnozę dotyczącą pracy samej sieci. Dają one co najwyżej podstawę do kontroli poprawności pracy i usuwania ewentualnych usterek samej warstwy transportowej. Do uzyskania interesujących administratora sieci komputerowej informacji koniecznych jest wiele zabiegów, które oprócz samej czasochłonności wymagałyby konkretnej wiedzy z zakresu wykorzystania funkcji API biblioteki WINSOCK w celu interpretacji parametrów wywołań i zwracanych rezultatów.

2.1. Specyfikacja Winsock

Specyfikacja WINSOCK (skrót od *Windows Sockets*) definiuje interfejs programowy (ang. *API - Application Programming Interface*), umożliwiający wykorzystanie usług sieciowych

TCP/IP w środowisku Windows™. Ze względu na numer wersji tej specyfikacji oraz środowisko, dla którego jest przeznaczona, biblioteka ta nosi różne nazwy[1].

Tabela 1

Nazwy biblioteki WINSOCK

	WINSOCK 1.1	WINSOCK V 2.0
Wersja 16-bitowa	WINSOCK.DLL	WS2-16.DLL
Wersja 32-bitowa	WSOCK32.DLL	WS2-32.DLL

Obserwując rozwój rynku 32-bitowych systemów Windows™ oraz oprogramowania zgodnego z WINSOCK można stwierdzić, że w tej chwili praktyczne znaczenie mają 32-bitowe implementacje biblioteki w wersji 1.1, należy zaś przypuszczać, iż WINSOCK v 2.0 w wersji 16-bitowej (czyli dla Windows 3.1 i 3.11) prawdopodobnie nie będzie nawet implementowany. Specyfikacja ta gwarantuje, że stosując się do niej uzyskujemy aplikację niezależną od producenta biblioteki WINSOCK i vice versa - można stosować jedną taką bibliotekę do komunikowania się aplikacji stworzonych przez różnych producentów.

Wersja 1.1 udostępniona w styczniu 1993 roku jest zgodna z wytycznymi i strukturą nakreśloną w pierwowzorze. Wprowadziła zmiany, w stosunku do wersji 1.0, tylko tam, gdzie było to absolutnie konieczne zgodnie z doświadczeniami wielu firm dotychczas implementujących bibliotekę WINSOCK.

Od 1995 roku prowadzone są prace nad nowszą wersją specyfikacji Winsock - 2.0. W czasie pisania tego opracowania wciąż jeszcze trwają. Wprawdzie dostępne są już pierwsze nowe wersje, jednak z zastrzeżeniem prawa do wprowadzania zmian bez uprzedzenia. W tej sytuacji, przy braku ostatecznej wersji specyfikacji, przy realizacji oprogramowania będącego tematem niniejszego opracowania nie została ona uwzględniona. Należy jednak podkreślić, że:

- jest ona następczynią wersji 1.1 formalizującą wykorzystanie wielu zestawów protokołów takich jak: DECNet, IPX/SPX, itp., pozwalającą na ich równoczesną koegzystencję, dodającą nowe cechy funkcjonalne, a co najważniejsze zachowującą pełną zgodność z wersją 1.1,
- wprowadza możliwość monitorowania i debuggowania pracy biblioteki (dzięki odpowiedniej jej wersji i dodatkowemu modułowi - bibliotece DT_DLL.DLL), co może mieć znaczenie przy rozwijaniu niniejszej koncepcji w przyszłości.

[1] Dla uproszczenia dalszych rozważań, kiedy użyta zostanie nazwa WINSOCK lub WINSOCK.DLL, odnosić się będzie ona mogła do dowolnej z implementacji, niezależnie od jej wersji (1.1 czy 2.0) i środowiska (16 czy 32-bitowe). Wersje 32-bitowe są przeznaczone wyłącznie dla środowisk Windows 95 i NT.

2.2. Techniczne aspekty monitora wywołań funkcji API w środowiskach Windows™ 95 i Windows NT

Najbardziej skutecznym i niezawodnym, bo realizowanym sprzętowo, sposobem śledzenia odwołań do pamięci operacyjnej, w tym oczywiście wywołań funkcji API, jest zestaw przerwań dostępnych w procesorach x86. Mniej wygodną metodą jest operowanie na *Module Database* modułów w pamięci, tak aby najpierw wywoływana była nasza funkcja rejestrująca, a dopiero później właściwa realizująca żądaną usługę. Jednak najbardziej eleganckie i najmniej uciążliwe zdaje się być zastąpienie biblioteki WINSOCK. Koncepcje ostatnia i przedostatnia są podobne do siebie pod względem działania rejestratora zdarzeń, jednak ta ostatnia nie ingeruje tak głęboko w system operacyjny, przez co jej funkcjonowanie jest mniej uzależnione od wprowadzanych w przyszłości w nim zmian.

2.3. Przerwania sprzętowe, łańcuch obsługi przerwania 03H

Warunki automatycznego generowania przez procesor przerwania 01h określa zawartość jego czterech rejestrów przeznaczonych wyłącznie do tego celu. Należy do nich wpisać adresy komórek w pamięci, przed wykonaniem zawartości których następuje jego generacja. Niedogodnością takiego rozwiązania jest ograniczenie do 4 wybranych lokalizacji w PaO, co w kontekście niniejszej pracy jest środkiem niewystarczającym. Specyfikacja WINSOCK w wersji 1.1 obejmuje 47 różnych funkcji i pomimo że znaczących dla procesu monitorowania jest zaledwie 8, to i tak jest liczbą zbyt dużą jak na możliwości wynikające z wykorzystania przerwania 01h procesorów x86.

Przerwanie 03h generowane jest w sposób programowy - należy do śledzonego kodu wstawić kod instrukcji *INT 3*, aby po jej wykonaniu procesor przeszedł w tryb jego obsługi. Można wówczas przeprowadzić rejestrację charakteru wywoływanej funkcji, argumentów wywołania, a następnie należy odtworzyć zachowane wcześniej instrukcje, których miejsce zastąpiła instrukcja *INT3* i przekazać im sterowanie, w celu normalnej kontynuacji działania.

Oba ww. rozwiązania wiążą się ze znaczącymi utrudnieniami. Gdybyśmy chcieli kontrolować rezultaty wywołań funkcji, należałoby dokonać analizy kodu w postaci binarnej identyfikując w nim prologi i epilogi funkcji. Ponadto kontrola parametrów wywołania i ewentualnie wyników pociągałaby za sobą konieczność znajomości organizacji stosu oraz postaci przekazywanych danych.

W środowisku Windows 95 i NT obsługa przerwania 03h wymaga „zejścia" na poziom pierścienia „0" systemu operacyjnego, czyli zastosowania wirtualnych sterowników VxD oraz wpięcia się do istniejącego łańcucha obsługi tego przerwania.

2.4. Operowanie na Module Database modułów

Dla potrzeb systemów Windows, Microsoft zdefiniował dotychczas 2 formaty plików z wyróżnionymi blokami: kodu, danych i zasobów, które wykorzystywane są do opisu budowy bibliotek DLL (również w aplikacjach).

W wypadku Windows 3.x jest to format NE (ang. *New Executable*), w którym oprócz ww. bloków występuje informacja o podziale na segmenty, co ma związek z organizacją pamięci w środowisku Windows w trybie 16-bitowego adresowania pamięci.

Dla systemów Windows 95 i NT jest to format PE (ang *Portable Executable*) wywodzący się ze specyfikacji COFF (ang. *Common Object File Format*) powszechnie wykorzystywanej w systemach UNIX.

3. Koncepcja funkcjonowania monitora

Po załadowaniu biblioteki WINSOCK do pamięci należy wykonać kopię biblioteki WINSOCK. Następnie modyfikujemy kod prologów wszystkich eksportowanych funkcji właściwego modułu WINSOCK, tak aby jedynym ich zadaniem było wywołanie odpowiadających im funkcji w monitorze, np. w wyniku wywołania funkcji *recv()* oryginalnego modułu WINSOCK zostanie wywołana funkcja *new_recv()* z modułu monitora, która po zarejestrowaniu argumentów wywołania przekazuje sterowanie do kopii kodu właściwego modułu - funkcji *recv()*. Po powrocie z niego następuje rejestracja zwróconego rezultatu, powrót do wywołującej funkcji *recv()*, której zadaniem jest powrót do kodu wywołującego ją modułu. Można byłoby również dokonywać bezpośredniego powrotu z funkcji *new_recv()* po pobraniu adresu powrotu ze stosu.

W trakcie wykonywania wszystkich tych operacji należałoby również przed każdym wywołaniem i powrotem z funkcji zatroszczyć się o zawartość stosu.

Zaletą takiego rozwiązania jest dostęp do funkcji, których istnienie nie było wiadome w trakcie tworzenia monitora, ponieważ modyfikacja dotyczyłaby tylko wybranych pozycji, pozostałe z nich funkcjonowałyby niezmiennie.

Modyfikacja kodu modułu jest operacją niezbyt skomplikowaną w środowiskach Windows 3.x, ze względu na wspólną przestrzeń adresową wszystkich procesów, gdzie istnieje zawsze jedna kopia kodu bibliotek DLL (również danych - w odróżnieniu od aplikacji). Modyfikując kod w jednym procesie spowodujemy, że zmiany będą widoczne w pozostałych używających tego modułu. Nie ma również problemów z lokalizacją w pamięci kolejno wczytywanych modułów. Gdy wykorzysta się funkcje systemowe oraz usługi dostępne w module

TOOLHELP.DLL, zlokalizowanie kodu w pamięci i jego modyfikacja nie nastręcza problemów.

Sytuacja z dostępem do zawartości pamięci operacyjnej uległa diametralnej zmianie w systemach Windows 95 i NT, gdzie każdy 32-bitowy proces posiada indywidualną przestrzeń adresową.

Dynamicznie łączone biblioteki (DLL) są tworzone z domyślnym adresem bazowym, pod który powinny zostać załadowane. Jeśli jednak z jakiegoś powodu DLL nie może zostać załadowany pod ten adres, zostanie on umieszczony w innym miejscu. W takiej sytuacji kod musi zostać skopiowany do innej fizycznej ramki, ponieważ korekty instrukcji skoków w DLL są zapisywane jako specyficzne lokalizacje w stronach biblioteki. W wyniku korekty następuje próba modyfikacji dotychczas „wspólnej" pamięci, co powoduje zadziałanie mechanizmu *Copy on Write* opisanego poniżej. Jest to istotny fakt przy decydowaniu o wyborze takiej metody monitorowania, ponieważ kod samego monitora oraz kopia kodu oryginalnej biblioteki WINSOCK musiałyby znajdować się w tym samym miejscu w przestrzeni adresowej wszystkich monitorowanych procesów, aby można było je wskazać każdemu z nich. Jeśli tak nie jest, to zmodyfikowany kod mógłby odwoływać się do nieokreślonych obiektów w pamięci.

Systemy Windows 95 i NT stosują ochronę zawartości pamięci zgodnie z zasadą *Copy on Write*, czyli „kopia w wypadku zapisu". Aplikacja może załadować do swojej wirtualnej pamięci jakiś moduł, np. kod DLL. Inny proces mógłby załadować ten sam kod do swojej wirtualnej pamięci. Dopóki żaden z procesów nie dokonuje zapisu do tej przestrzeni, mogą one współdzielić te same fizyczne strony. Jeśli jednak któryś z nich dokona zapisu do tego obszaru pamięci (oznaczonego w wypadku DLL atrybutem *Copy on Write*), ramka fizycznej strony zostanie skopiowana w inne miejsce pamięci fizycznej oraz nastąpi korekta mapowania pamięci wirtualnej piszącego procesu.

W takiej sytuacji należałoby przeprowadzić modyfikację dla każdego procesu (każdej kopii biblioteki) osobno, śledzić ładowanie modułów do pamięci i ich usuwanie. Ze względu na istniejące systemy zabezpieczeń możemy nie mieć takiej możliwości, jeśli interesujące nas zadanie zablokuje dostęp do swoich obiektów. Utracimy wówczas część informacji, być może znaczącą, co stawia pod znakiem zapytania całą ideę monitorowania.

3.1. Zastąpienie biblioteki DLL

Ideę monitorowania wywołań funkcji API przez zamianę biblioteki ilustruje rysunek 1. Nazwa oryginalnej biblioteki WSOCK32.DLL zostaje zmieniona np. na WSOCK31.DLL. Należy zmodyfikować nie tylko nazwę pliku dyskowego, ale również identyfikator zawarty w wewnętrznych strukturach opisujący ją już po załadowaniu do środowiska WINDOWS. Po

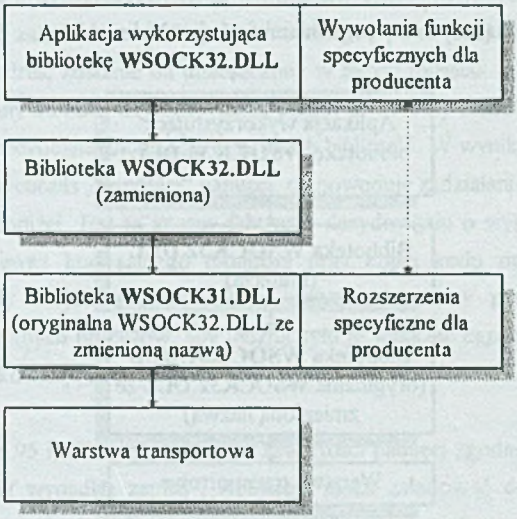
dokonaniu takiej operacji nasza „udawana" biblioteka WINSOCK ma za zadanie jedynie rejestrować wywołania funkcji API, przeprowadzać obróbkę wstępną tak gromadzonych danych, wywoływać odpowiednie funkcje właściwego modułu (już funkcjonującego pod zmienioną nazwą WSOCK31) i zwracać rezultaty tych wywołań. Z punktu widzenia aplikacji wykorzystujących bibliotekę WINSOCK sytuacja nie ulega żadnej zmianie - nadal mogą z niej korzystać na takich samych zasadach, zaś w czasie rzeczywistym następuje rejestracja wszystkich zdarzeń związanych z wywołaniami funkcji z biblioteki.

Rys. 1. Idea zamiany biblioteki DLL
Fig. 1. Idea of substituting DLL library

Proponowana zamiana biblioteki WINSOCK może wiązać się jednak z możliwością wystąpienia braku dostępu do specyficznych dla danego producenta funkcji, które są dodawane do biblioteki (rozszerzenia producenta) - patrz rysunek 2. Jednak w takiej sytuacji (powołując się na specyfikację WINSOCK) aplikacja ta jest uznawana za nie w pełni zgodną ze standardem WINSOCK. Pomimo iż np. Microsoft oferuje w swoich implementacjach bogactwo funkcji uzupełniających (patrz tabela poniżej), to w publikowanych przez siebie dokumentach (*Knowledge Base*) sugeruje, aby nie wykorzystywać tych rozszerzeń, gdyż utracimy możliwość wykorzystania dowolnej biblioteki WINSOCK, a zwiążemy się jedynie z jednym produktem.

W wypadku biblioteki WSOCK32.DLL autorstwa Microsoft pojawił się znaczący problem nie mający związku z przytoczonymi wyżej możliwymi przeszkodami. W trakcie analizy funkcjonowania tego modułu okazało się, że system operacyjny Windows 95 jest bardzo głęboko z nim powiązany. Kod jądra KERNEL32.DLL zawiera niejawne odwołania do biblioteki WSOCK32.DLL. Po zamianie nazwy modułu i zastąpieniu pierwowzoru przez rejestrator, system nadal „wie", że obecnie skonfigurowany do wykorzystania jest moduł

WSOCK32.DLL autorstwa Microsoft, więc nadal próbuje odwoływać się do własnych funkcji. W takiej sytuacji, jako że moduł jest identyfikowany przez nazwę, system nie potrafi automatycznie zlokalizować wymaganych funkcji i dlatego takie rozwiązanie musi zostać uzupełnione o uaktualnienie nazwy również w module jądra. Ponadto sama biblioteka lub inne związane z nią moduły wykorzystują dwa klucze w Registry.



Rys. 2. Odstępstwa od specyfikacji mogą spowodować brak dostępu do nieznanych funkcji
Fig. 2. Loss of access of unspecified functions

Wobec tego w trakcie instalacji należy również zmodyfikować wszystkie ww. obiekty. Mankamentem takiego rozwiązania może się okazać ponadto brak możliwości rejestrowania przesyłów wykonywanych z wykorzystaniem dodatkowych funkcji (są to prawdopodobnie *Arecv()* i *Asend()*), z których mogą korzystać aplikacje producenta.

Podobne problemy nie wystąpiły w wypadku innych komercyjnych produktów: NetManage - Internet Chameleon v 5.0., FTP - OnNet 2.0, Sunsoft - PC-NFS^{PRO} 2.0.

4. Wybór modelu rozwiązania

Ze względu na dużą efektywność i przezroczystość w połączeniu z niezbyt skomplikowaną metodą realizacji, a co najważniejsze, najmniejszy stopień ingerencji w funkcjonowanie systemu operacyjnego wybrana, zrealizowana i przetestowana została metoda wykorzystująca zastąpienie biblioteki DLL. Przy realizacji wykorzystano koncepcję mapowania plików w przestrzeni adresowej oraz semafory do synchronizacji współdziałania procesów.

4.1. Podział na moduły funkcjonalne

W trakcie realizacji projektu wyróżniono dwa moduły funkcjonalne:

- Rejestrator zdarzeń w postaci 32-bitowej biblioteki WSOCK32.DLL.
- Moduł interpretujący i wizualizujący zgromadzone dane.

4.1.1. Konstrukcja rejestratora

Moduł rejestratora zdarzeń ma za zadanie gromadzić opisy wszystkich interesujących, z punktu widzenia oceny funkcjonowania sieci, zdarzenia w postaci wywołań funkcji API biblioteki WINSOCK oraz zwracanych rezultatów, a następnie umieszczać je w archiwum zrealizowanym w postaci pliku, który w trakcie działania jest mapowany w przestrzeni adresowej procesu. Moduł uruchamiany jest jedynie w sytuacji, kiedy któraś z aplikacji wykorzystuje transport TCP/IP wywołując funkcje z biblioteki WSOCK32.DLL. Do rejestrowania zostały wybrane niektóre z usług API wpływające bezpośrednio na natężenie ruchu w sieci komputerowej: *recv(), recvfrom(), send(), sendto(), gethostbyaddr(), WSAgethostbyaddr(), gethostbyname(), WSAgethostbyaddr()*.

W wypadku dwu ostatnich grup funkcji nie jest możliwe ocenienie wpływu ich wywołań na obciążenie sieci komputerowej. Możliwe jest, iż implementacje biblioteki WINSOCK buforują informacje związane z konwersją i tylko pierwsze wywołanie wiąże się z jakimś przesyłem, kolejne zaś to odczyty z lokalnego bufora. Aby mieć możliwość rzetelnej oceny, należałoby znać szczegóły ich implementacji, co jest raczej trudne do wykonania ze względu na brak dostępu do tekstu źródłowego.

4.2. Budowa modułu wizualizacji

Drugi moduł, którego zadaniem jest obróbka zgromadzonych i napływających na bieżąco danych, udostępnia szereg różnorodnych statystyk i zestawień. Po uruchomieniu aplikacji następuje wypełnienie buforów przetworzonymi już danymi - rekordy o identycznych kluczach (np. IP-adresach i numerach portów) są grupowane, a informacje w nich zawarte (np. ilość przesyłanych danych) - sumowane. Następnie ich zawartość zostaje posortowana wg kryteriów zależnych od rodzaju statystyki. W trakcie działania Monitora następuje okresowe uaktualnianie ich zawartości. W przypadku dynamicznego przydzielania adresów (protokoły BOOTP, DHCP) analiza pracy urządzeń lokalnych oraz identyfikacja łączących się z siecią komputerów odległych są niestety pozbawione sensu. Nie jest to jednak niedoskonałość monitora, ale wynik sposobu funkcjonowania protokołów TCP/IP.

4.3. Opis oferowanych statystyk

Zestaw dostępnych statystyk w module analizy i wizualizacji obejmuje:

- Prezentację i wydruk ilości wysłanych i odebranych informacji przypadających na poszczególne adresy odległe i lokalne.
- Prezentację i wydruk ilości wysłanych i odbieranych informacji przypadających na poszczególne porty urządzeń odległych i lokalnych.
- Prezentację i wydruk czasów trwania połączeń z wyszczególnieniem adresów lub portów.
- Wykaz i wydruk wykorzystywanych nazw domenowych wraz z licznikami ich użyć.
- Wykaz i wydruk zarejestrowanych kodów błędów mających związek z funkcjonowaniem sieci komputerowej.
- Zawartość i wydruk zawartości pomocniczych tablic statystyk wykorzystywanych do tworzenia pozostałych zestawień.
- Rozkład częstotliwości występowania zdarzeń w czasie z podziałem na dni miesiąca i pory tych dni (wraz z wydrukiem).

Prezentowane statystyki pozwalają na efektywną ocenę pracy sieci i wspomaganie podejmowania decyzji z nią związanych, np.:

- Wybór fragmentu sieci, w którym wskazane jest zwiększenie przepustowości łączy.
- Modernizacja komputera, na którym odbywa się zdalna praca, niepolepszanie łącza w przypadku intensywnej pracy w wykorzystaniem portu 23 (Telnet).
- Wskazanie użytkowników odpowiedzialnych za dużą częstotliwość transferów plików - port 21. Próba zlecania systemowi tych operacji na późniejsze godziny.
- Planowanie konserwacji sprzętu w oparciu o informacje o najmniejszym wykorzystaniu systemu lub poszczególnych urządzeń.
- Uzupełnianie zawartości tablic serwerów DNS w celu redukcji ruchu na zewnątrz.
- Analiza występujących i zarejestrowanych błędów w pracy sieci i oprogramowania.

5. Podsumowanie

Coraz szybszy rozwój sieci komputerowych, zwłaszcza sieci Internet, powoduje coraz częstszą potrzebę stałego nadzoru na siecią. Jak podaje w swoich badaniach firma Infonetics, 94% firm połączy swoje lokalne oddziały z centralą za pomocą sieci do roku 1997. W celu utrzymania najwyższych parametrów pracy sieci (przepustowość, szybkość, niezawodność) należy dysponować analitycznymi danymi pozwalającymi na odpowiednie "strojenie" sieci.

Zaprezentowany monitor sieci TCP/IP w środowisku Windows umożliwia rejestrację i wizualizację zdarzeń mających bezpośredni wpływ na obciążenie pracy tejże sieci, pozwala na równoczesną rejestrację zdarzeń, ich wizualizację z uwzględnieniem kilku różnych kryteriów równocześnie połączoną z cyklicznym uaktualnianiem, generowanie raportów i niczym nie różniącą się od dotychczasowej pracę aplikacji wykorzystujących bibliotekę WSOCK32.DLL.

Należy podkreślić, iż rozwiązanie to przygotowane jest również do zastosowania przy:

- rozliczaniu ruchu w sieci,
- blokowaniu niepożądanej komunikacji (system typu Firewall)

LITERATURA

[1] Windows 95 User's Guide, Microsoft Corporation 1995.
[2] Windows NT System Guide, Microsoft Corporation 1995.
[3] Microsoft Windows 95 Resource Kit, Microsoft Press 1995.
[4] Microsoft Windows NT 4.0 Serwer Reviews Guide 1996.
[5] Winsock 1.1 i 2.0 Specification.

Recenzent: Dr inż. Ryszard Winiarczyk

Wpłynęło do Redakcji 20 listopada 1996 r.

Abstract

One of the biggest technical successes that became a progress, modernity and integration symbol is Internet - the global network. Giving a huge possibilities of data exchange causes users to bear with one problem - there is never sufficient throughoutput available to them. Using any of offered by the net facilities is strongly related with data flow generating.

How could network administrators solve the problems not having enough money to modernize existing network structures. The answer could be monitoring and analysing of network traffic and errors occurrences, supporting decisions making.

One of the most popular 32-bit operating systems are Windows 95 and NT and we should expect, that most of traffic in all networks could be generated by them. Communicating with each other with the most popular protocol suite - TCP/IP, use an implementation of Winsock library in version 1.1 or higher.

If we would like to monitor the data flow we should monitor the use of that library. There are three methods of doing that: hardware and software interrupts, modifying the *Module Database* of a library or substituting the whole module.

The first method is the most complicated one and demands using Virtual Driver on an "0" ring of a system. The second one, considering methods of memory access in Windows environments, appears to be less complicated, but the third one gives us a stright and clear way to monitor what we expect, not concentrating us mainly on technical problems.

Substituting a module - see figure 1 - allows a transparent data flow description registration. There could appear one main problem - see figure 2 - we could probably loose an access to undocumented functions available in substituted module. But - considering a specification - that library is not completely compliant with it. And when application demands a Winsock module it should work without those extensions.

As an result there has been developed a monitor model based on the above presented foundations, that is also able to present registered results in real time allowing administrators undertake critical decisions.