POLITECHNIKA WROCŁAWSKA, WYDZIAŁ INFORMATYKI I ZARZĄDZANIA

Kamil Czarnecki

Algorytmy, projekt i realizacja systemu do zarządzania wielokomputerowymi serwerami WWW

promotor: Prof. Dr hab. inż. Leszek Borzemski
konsultant: Mgr inż. Radosław Porczyński

Wrocław, 2001

pracę dedykuję Moim Rodzicom

Spis treści

Rozdział 1

Wprowadzenie. Cel pracy

Celem pracy jest zaprojektowanie i realizacja systemu do zarządzania wielokomputerowymi serwerami WWW oraz przedstawienie całokształtu tematyki w zakresie równoważenia obciążeń systemów wielokomputerowych i problematyki z tym związanej.

Działanie ogólnoświatowej sieci rozległej – Internetu, rozumiane jest jako „luźno zorganizowana międzynarodowa współpraca autonomicznych, połączonych ze sobą sieci komputerowych, w oparciu o komunikację host–to–host poprzez dobrowolną przynależność do otwartych protokołów i procedur zdefiniowanych w dokumentach Internet Standards, RFC 1310, RFC1311 i RFC 1312".

Początków Internetu należy szukać w projektach realizowanych na zlecenie Departamentu Obrony USA przez agencję ARPA (ang. *Advanced Research Projects Agency*) oraz jej następczynię – agencję DARPA (ang. *Defense Advanced Research Projects Agency*). Przed tymi instytucjami postawiono zadanie opracowania standardów umożliwiających budowę rozległej sieci komunikacyjnej odpornej na unicestwienie nawet poprzez atak nuklearny znacznej części jej węzłów [**?**]. W wyniku prowadzonych prac w roku 1971 przekazana zostaje do użytku sieć ARPANET. Stosowane w niej protokoły transmisji nie zapewniały jednak wystarczającej przepustowości dla stale rosnącej liczby użytkowników i już w roku 1973 pojawiają się propozycje pierwszych protokołów z rodziny TCP/IP – *de facto* dzisiejszego standardu transmisji internetowych. W roku 1981 na bazie ARPANET–u utworzona zostaje CSN (ang. *Computer Science Newtwork*) wykorzystywana wspólnie przez abonentów wojskowych i cywilnych. W roku 1982 protokoły TCP/IP wypierają całkowicie starsze protokoły (np. protokół NCP), a do CSN średnio co 20 dni podłączany jest nowy komputer. W roku 1984 następuje podział sieci na część wojskową – MILNET i cywilną, która zachowuje tradycyjną nazwę ARPANET. W miarę jak zaczyna przekraczać ona granice USA i w miarę przybywania abonentów komercyjnych zaczyna pojawiać się nazwa Internet. W roku 1990 decyzją rządu USA szkielet sieci ARPANET składający się z czterech węzłów komutacyjnch (Los Angeles, Santa Barbara, Uniwersytet Stanforda i Uniwersytet Utah) zbudowanych w oparciu o minikomputery Honeywell 316 zostaje uznany za przestarzały i przeznaczony do rozbiórki. Zastępuje go sieć NSFNET, która do dziś jest

szkieletową siecią amerykańskiej części Internetu.

Kluczową ze względu na treść tej pracy datę w historii Internetu jest rok 1990. Wtedy to w Europejskim Ośrodku Badań Jądrowych CERN w Genewie został opracowany i uruchomiony pierwszy serwer WWW (ang. *World Wide Web*). WWW jest systemem hipertekstowym tzn. przechowującym dane w postaci sieci dokumentów (plików) połączonych poprzez odsyłacze. Sam dokument może zawierać tekst, grafikę, dźwięk, animacje, sekwencje video i inne rodzaje danych. Elementy wskazywane przez odsyłacze mogą znajdować się w tym samym dokumencie lub być innymi dokumentami na tym samym lub innym serwerze, co dokument zawierający dany odsyłacz. Obecnie WWW to zbiór dziesiątków tysięcy, połączonych siecią odsyłaczy, serwerów w wielu krajach. Każdy serwer udostępnia użytkownikom zgromadzone na nim dokumenty o różnorodnej treści informacyjnej, reklamowej, rozrywkowej, handlowej (e–banki, sklepy internetowe) itp. Narzędziem umożliwiającym zwykłemu użytkownikowi Internetu korzystanie z zasobów WWW jest przeglądarka (ang. *browser*). Komunikacja przeglądarki z serwerem WWW odbywa się w architekturze klient–serwer i jest zawsze inicjowana przez zapytania ze strony klienta (przeglądarki). Odpowiadając na zapytania przeglądarki serwer WWW przesyła do niej żądane dokumenty z wykorzystaniem protokołu HTTP (ang. *Hyper Text Transfer Protocol*). Najczęściej stosowanym językiem opisu dokumentu hipertekstowego jest HTML (ang. *Hyper Text Mark up Language*). Przeglądarka interpretując otrzymany plik HTML potrafi odpowiednio rozmieścić na ekranie użytkownika tekst, grafikę i odsyłacze składające się na dokument. Pliki HTML mają charakter statyczny – służą do prezentacji wcześniej przygotowanych danych. Specyfikacja HTML jest jednak bardzo elastyczna – do plików HTML można włączać skompilowane do postaci kodów bajtowych programy języka Java, skrypty JavaScript, Visual Basic–a czy kontrolki ActiveX, w ten sposób serwer WWW powoduje uruchomienie w komputerze użytkownika programów przesłanych w dokumencie, co uatrakcyjnia wygląd dokumentu i umożliwia pewien stopień dialogu z użytkownikiem. Dla zadań takich jak np. wyszukiwanie danych do pliku HTML dołączać można odwołania do programów (skryptów) CGI (ang. *Common Gateway Interface*). Program napisany zgodnie z normą CGI wykonywany jest w tym samym komputerze, w którym działa proces serwera WWW – w takiej sytuacji komputer ten staje się serwerem aplikacji (może również prezentować dane wydobywane z baz danych).

Nie sposób mówić o serwerach WWW w oderwaniu od takich usług internetowych jak Gopher, czy serwery FTP, gdyż wiele dokumentów hipertekstowych zawiera odsyłacze inicjujące pobieranie plików z serwerów FTP lub dokonujących przełączenia do systemu Gopher (czysto tekstowego systemu informacyjnego).

Przedstawiana praca dotyczy problematyki rozwoju sieci Internet w zakresie burzliwie rozwijających się systemów informacyjnych opartych o WWW, a dokładniej możliwości budowy i implementacji wysokowydajnych systemów WWW opartych o architekturę wieloserwerową. Tradycyjny system WWW, składający się z pojedynczego komputera nie jest w stanie spełnić wymagań rosnącej liczby coraz bardziej wybrednych klientów (zwiększanie procesorów, pamięci operacyjne i dyskowej w platformach jednoserwero-

wych ma swoje granice). Przeciętny użytkownik „pajęczyny" oczekuje w miarę szybkiego, ale przede wszystkim zawsze dostępnego serwisu internetowego. Mając na uwadze fakt, że systemy WWW stanowią od pewnego czasu znakomitą platformę do prowadzenia handlu elektronicznego – dostępność witryn jest niezbędna do utrzymania się na rynku firm prowadzących tego rodzaju usługi. Jedynym, jak się wydaje, rozwiązaniem jest tworzenie systemów WWW budowanych na bazie wieloserwerowej – tylko takie rozwiązanie może zapewnić ciągłą dostępność do zasobów WWW (ich główną zaletą jest możliwość stopniowej i teoretycznie niograniczonej rozbudowy).

Już w roku 1995 pakiety HTTP stanowiły 36% wszystkich pakietów przesyłanych w sieci szkieletowej NSFNET [**?**] i udział ten wykazywał stałą tendencję wzrostową. Obecnie głównym motorem rozwoju „światowej pajęczyny" są wszelkiego rodzaju e–biznesy (platformy B2B, B2C i inne), co wiąże się z ogromną integracją tradycyjnego statycznego HTML–a (lub XML–a) z elementami dynamicznymi zmieniającymi się w czasie. Zwiększa to już i tak niemałe trudności w optymalnym konstruowaniu serwerów WWW. W związku z problemami związanymi z potężnym przyrostem internautów oraz komplikacjami w ciągłym rozwijaniu pojedynczych komputerów będących serwerami WWW, a także wiązaniu ze sobą różnorakich usług internetowych – stale zwiększającym się uznaniem (oraz znakomitym stosunkiem wydajność/cena) cieszą się systemy rozproszone:

klastry webowe – serwery WWW rozproszone lokalnie tzn. funkcjonujące w obrębie sieci lokalnej;

rozproszone serwery webowe – rozproszone geograficznie serwery webowe pomiędzy którymi istnieje mechanizm szeregowania;

rozproszone klastry webowe – jest to szczególny przypadek rozmieszczenia serwerów webowych – stanowi kombinację globalnego i lokalnego rozproszenia.

Problemem w ich wykorzystaniu jest odpowiednie skierowanie klientów do najlepszego serwisu tzn. takiego, który bez względu na obciążenie jest osiągalny dla klienta i zwraca mu odpowiedź możliwie najszybciej. Aby zrealizować tego typu architekturę konieczne jest stworzenie odpowiednich mechanizmów szeregowania zapytań klienckich, algorytmów szeregowania wyznaczającego najlepszy serwer oraz urządzenia, w którym jest zaimplementowany mechanizm i algorytm szeregowania. W zależności od umiejscowienia jednostki szeregującej zapytania od klientów można wyróżnić trzy grupy metod:

- podejście: po stronie klienta (wymagające modyfikacji po stronie klienta – np.: ingerencji w budowę przeglądarki);
- podejście z wykożystaniem niezależnego węzła dystrybuującego zapytania od klientów (polegające na modyfikacji lub przekazywaniu pakietów kierowanych do systemu konkretnym serwerom realizującym zlecenie – np.: komputer z zainstalowanym oprogramowaniem SecureWay Network Dispatcher firmy IBM);

- podejście: po stronie serwera (wymagające modyfikacji po stronie serwera)

Szczegółowo, zarówno mechanizmy szeregowania zapytań, algorytmy szeregowania jak i urządzenia, oraz sposób implementacji tak mechanizmów jak i algorytmów zostanie szczegółowo omówiony w dalszej części tej pracy (patrz: rozdział 3 i 4)

Jako przykład systemu o większej wydajności i dostępności zaimplementowane zostanie rozwiązanie oparte o architekturę wieloserwerową w oparciu o komputery IBM RS/6000 pracujące pod kontrolą systemu operacyjnego AIX oraz maszyny typu PC pracujące pod kontrolą MS Windows NT z wykorzystaniem profesjonalnego oprogramowania równoważącego obciążenie – pakietu IBM SecureWay Nework Dispatcher.

Celem pracy jest prezentacja algorytmów i rozwiązań służących do zarządzania rozproszonymi systemami WWW oraz przykład implementacji takiego rozwiązania. Omawiana instalacja serwerowa została zestawiona w laboratorium Instytutu Sterowania i Techniki Systemów Politechniki Wrocławskiej.

W dalszej części pracy zostaną przedstawione protkoły TCP/IP, których znajomość jest konieczna do zrozumienia treści zawartych w części praktycznej. Ten krótki opis obejmie IP, TCP, UDP, HTTP, FTP, SMTP, SNMP, SSL, DNS, oraz adresowanie URL. Po opisie protokołów zostanie szczegółowo przybliżona architektura klient–serwer, struktura WWW oraz klasyfikacja i opis architektur rozproszonych serwerów WWW. Kolejny rozdział wyjaśnia czym jest i do czego jest potrzebne zarządzanie wielkomputerowymi systemami WWW oraz w jakich sytuacjach się to zarządzanie stosuje. Ostatnim punktem tej pracy jest opis konfiguracji przykładowego systemu WWW wykorzystującego oprogramowanie IBM Network Dispatcher[1].

[1] Aktualnie IBM Network Dispatcher jest elementem oprogramowania o nazwie IBM WebSphere Edge Server

Rozdział 2

Protokoły komunikacyjne (sieciowe)

W rozdziale tym przedstawiona zostanie architektura ISO/OSI stosu protokołów, dzięki którym wydajna i bezpieczna transmisja w sieci nie byłaby w ogóle możliwa: od protokołów TCP/IP, poprzez SSL do protokołów FTP i HTTP. Opisany zostanie także sposób identyfikacji i komunikacji urządzeń i zasobów sieciowych na różnych warstwach czyli DNS i URL. Najwięcej uwagi poświęcone zostanie na opis protokołu HTTP i sposób działania WWW.

2.1 Wstęp

Podstawow urządzenia komunikacyjne składają się z elementów, które przesyłają bity z jednego miejsca do drugiego. Jednakże używanie „surowego" sprzętu do komunikacji jest podobne do programowania przez wrowadzanie zer i jedynek – jest to nieporęczne i niewygodne. Komputery podłączone do sieci są wyposażone w skomplikowane oprogramowanie udostępniające wygodny, wysokopoziomowy interfejs dla programów użytkownika. Oprogramowanie to obsługuje automatycznie większość niskopoziomowych szczegółów oraz problemów, co pozwala na łatwą komunikację między programami użytkowymi. W ten sposób większość programów użytkownika przy komunikacji polega na oprogramowaniu sieciowym i nie komunikuje się bezpośrednio ze sprzętem.

Wszystkie strony biorące udział w komunikacji muszą uzgodnić zbiór reguł używanych przy wymianie komunikatów. Zbiór reguł, które określają format komunikatów, oraz stosownych działań wymaganych dla każdego komunikatu nazywa się protokołem sieciowym.

2.2 TCP/IP a ISO/OSI

Aby sprawienie i efektywnie rozwiązać dowolne zadanie komunikacyjne oraz zaplanować zestaw protokołów stworzono model warstwowy [**?**]. Zestaw protokołów można tym sa-

mym związać z określoną warstwą [**?**].

Wcześnie w historii sieci *International Organization for Stadarization* (ISO) opracowała model 7–warstwowy. Model ISO/OSI jest standardem stanowiącym punkt odniesienia dla producentów oprogramowania i sprzętu sieciowego. Stosowanie się do zaleceń tego standardu zapewnić ma kompatybilność różnych produktów sieciowych, co implikuje możliwość swobodnego ich łączenia w tzw. system otwarty. Wyróżnić w nim można: warstwę aplikacji, prezentacji i sesji – nazywane warstwami górnymi, oraz warstwę transportową, sieciową, łącza danych i fizyczną – nazywane warstwami dolnymi. Warstwy górne zajmują się współpracą procesów wykonujących się w połączonych siecią komputerach. Warstwy dolne zapewniają prawidłową komunikację pomiędzy procesami. W każdej z warstw pracuje odpowiednie oprogramowanie realizujące funkcje protokołu danej warstwy. Protokoły poszczególnych warstw oferują określony rodzaj usług [**?**]:

warstwa aplikacji – określa sposób, w jaki dany program użytkowy korzysta z sieci np.: w wastwie tej działa protokół HTTP;

warstwa prezentacji – protokoły tej warstwy określają jak prezentować dane. Takie protokoły są potrzebne gdyż różne marki komputerów używają różnej wewnętrznej reprezentacji liczb i znaków – protokoły tej warstwy tłumaczą reprezentację na jednej maszynie na reprezentację na drugiej;

warstwa sesji – odpowiedzialna jest za synchronizację danych przesyłanych pomiędzy aplikacjami, dzięki udostępnianym w tej warstwie tzw. punktom synchronizacji aplikacje są w stanie wzajemnie rozpoznać stan zaawansowania w przetwarzaniu wymienianych danych, umożliwia to wykrycie takich sytuacji jak np. spowodowane awarią zaprzestanie przetwarzania przez stowarzyszoną aplikację;

warstwa transportowa – protokoły wastwy czwartej określają szczegóły obsługi połączenia – są jednymi z najbardziej skomplikowanych protokołów;

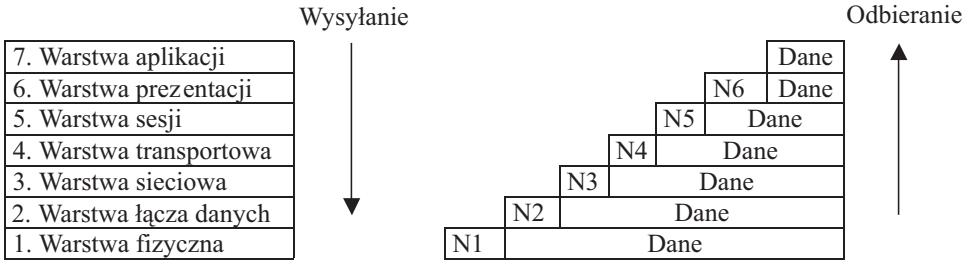
warstwa sieciowa – protokoły tej warstwy określają sposób przyznawania adresów oraz przekazywania pakietów w sieci;

warstwa łącza danych – protokoły tej warstwy określają sposób organizowania ramek i transmitowania ich przez sieć np.: opis formatu ramki, rozpychania bitów lub bajtów oraz obliczania sum kontrolnych;

warstwa fizyczna – realizuje mechaniczne, elektryczne, funkcjonalne i proceduralne aspekty transmisji danych, czyli definiuje medium transmisyjne takie jak np. kabel miedziany.

Każda z warstw udostępnia warstwie bezpośrednio wyższej zestaw funkcji realizujących zadania danej warstwy i korzysta z zestawu funkcji udostępnianych przez warstwę

bezpośrednio niższą. Nie jest możliwe by np. warstwa transportowa korzystała z funkcji warstwy łącza danych z pominięciem warstwy sieciowej. Porcję danych przeznaczonych do wysłania nazywa się pakietem. Wysyłany pakiet musi przejść kolejno przez wszystkie warstwy modelu ISO/OSI od warstwy aplikacji do warstwy fizycznej. W trakcie tej wędrówki pakiet obrasta w informacje kontrolne umożliwiające protokołowi danej warstwy zarządzanie danymi i połączeniami, tę informację kontrolną nazywamy nagłówkiem lub końcówką w zależności od tego, czy jest dodawana na początku, czy na końcu danych. Nagłówek warstwy wyższej traktowany jest w warstwie niższej jako część danych i nie podlega tam żadnym zmianom. Pakiet odbierany odbywa drogę odwrotną – od warstwy fizycznej do warstwy aplikacji. W trakcie tej drogi protokoły poszczególnych warstw usuwają swoje nagłówki, tak aby do aplikacji docelowej pakiet dotarł w takiej postaci, w jakiej został wysłany. Poniższy rysunek schematycznie przedstawia opisany proces.

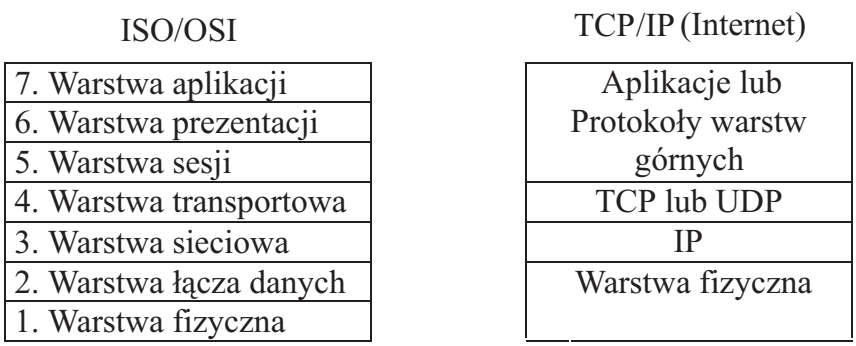


Rysunek 2.1: Proces nadawania i odbierania nagłówków transmitowanym danym

Protokoły poszczególnych warstw zarządzają danymi w specyficzny dla siebie sposób, mogą np. dokonać segmentacji, czyli podziału danych na mniejsze fragmenty przed przesłaniem ich do niższej warstwy. Każdy fragment otrzymuje wtedy odpowiedni nagłówek, który umożliwia analogicznemu protokołowi w komputerze adresata danych scalenie fragmentów podczas odbierania pakietu. Nagłówki umożliwiają również zarządzanie połączeniami np. multipleksowanie połączeń czyli wykorzystywanie jednego połączenia, ściślej – punktu dostarczania usług SAP (ang. *Service Access Point*) warstwy niższej, np. sieciowej do obsługi kilku połączeń nawiązanych w warstwie bezpośrednio wyższej (tu transportowej). Odpowiednia informacja zawarta w nagłówku warstwy wyższej umożliwia wtedy identyfikację odbiorcy danych. Model ISO/OSI definiuje dwa tryby komunikacji. Komunikacje połączeniową stosuje się gdy kluczowym wymaganiem jest niezawodność transmisji, musi być wtedy znany zarówno odbiorca jak i nadawca danych, a odbiorca potwierdza dotarcie do niego każdego fragmentu pakietu lub sygnalizuje konieczność retransmisji w przypadku błędu. Komunikacja bezpołączeniowa nie zapewnia niezawodności, dane wysyła się bez oczekiwania na potwierdzenie, a nadawca i odbiorca nie są znani, chyba że wynika to z zawartości pakietu, pakiety wysyłane w trybie bezpołączeniowym nazywamy

datagramami.

W praktyce nie istnieje rozwiązanie przemysłowe, które ściśle implementowałoby wszystkie warstwy modelu ISO/OSI, częstą praktyką jest np. łączenie warstw fizycznej i łącza danych w jedną warstwę. Również protokoły TCP/IP nie są w pełni kompatybilne z modelem ISO/OSI, nie są też jednak całkowicie niekompatybilne. Wzajemną relację architektury ISO/OSI i TCP/IP przedstawia poniższy rysunek [**?**].



Rysunek 2.2: Architektura ISO/OSI a TCP/IP

2.3 Protokół IP

Protokół IP (ang. *Internet Protocol*) działa w warstwie sieciowej i jest podstawowym protokołem rodziny TCP/IP – wszystkie dane napływające z wyższych warstw przesyłane są w sieci jako datagramy IP. Protokół ten świadczy zawodne, bezpołączeniowe usługi dostarczania pakietów. Przez zawodne rozumiemy tu taką właściwość, że IP nie daje gwarancji dostarczenia pakietu do punktu przeznaczenia, a wszelkie mechanizmy zapewnienia niezawodności transmisji muszą zostać zapewnione przez protokoły wyższych warstw (np. TCP). Termin bezpołączeniowy oznacza tu, że IP nie przechowuje żadnej informacji na temat prawidłowo przesyłanych datagramów, każdy z nich obsługiwany jest osobno, co w praktyce może oznaczać, że datagramy mogą docierać do adresata w innej kolejności, niż zostały wysłane. Wszystkie dane konieczne do poprawnej pracy IP zawarte są w jego nagłówku, który typowo (jeśli nie zawiera opcji) ma długość 20 bajtów i przedstawiony poniżej format [**?**].

Pole *długość nagłówka* określa długość nagłówka w 32–bitowych słowach i wraz z polem Długość całkowita datagramu służy do określenia miejsca w którym zaczynają się

<table>
<tr><td>Wersja
(4 bity)</td><td>Długość
nagłówka
(4 bity)</td><td>Typ usługi TOS
(8 bitów)</td><td colspan="2">Długość całkowita datagramu w bajtach
(16 bitów)</td></tr>
<tr><td colspan="3">Identyfikacja (16 bitów)</td><td>Znaczniki
(3 bity)</td><td>Przesunięcie fragmentu
(13 bitów)</td></tr>
<tr><td colspan="2">Czas życia TTL
(8 bitów)</td><td>Protokół
(8 bitów)</td><td colspan="2">Suma kontrolna nagłówka IP
(16 bitów)</td></tr>
<tr><td colspan="5">Adres źródłowy (32 bity)</td></tr>
<tr><td colspan="5">Adres docelowy (32 bity)</td></tr>
<tr><td colspan="5">Opcje dodatkowe (32 bity)</td></tr>
<tr><td colspan="5">Dane (0-65515 bajtów)</td></tr>
</table>

Rysunek 2.3: Format datagramu IP z wyróżnieniem nagłówka

właściwe dane. Całkowita długość datagramu ograniczona jest zwykle przez warstwę fizyczną np. dla sieci Ethernet jest to maksymalnie 1500 bajtów a wielkość tę nazywamy MTU (ang. *Maximum Transmission Unit*) danej sieci.

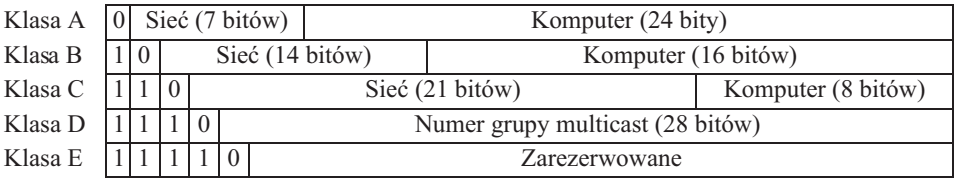
Pole *identyfikacja* zawiera unikalny numer dla każdego wysyłanego datagramu. Jeżeli przechodząc przez kolejne węzły sieci datagram musi zostać podzielony, to wartość ta jest kopiowana do wszystkich nagłówków IP powstających fragmentów i wraz z wartością pola Przesunięcie fragmentu umożliwia poprawne scalenie datagramu w punkcie docelowym. Pomocne są w tym również wartości bitów pola Znaczniki. Ustawienie jednego z nich wskazuje, że istnieje kilka fragmentów datagramu, ustawienie drugiego zabrania fragmentacji datagramu.

Pole *czas życia TTL* wyznacza maksymalną liczbę routerów przez które może przejść datagram (routerem nazywamy tu specjalizowane urządzenie lub odpowiednio oprogramowany komputer służące do przekazywania datagramów pomiędzy sieciami różniącymi się w warstwie fizycznej np. Ethernet i Token Ring, lecz pracującymi w oparciu o wspólny protokół warstwy sieciowej - IP). Wartość tego pola jest wielkością całkowitą zmniejszaną o jeden przez każdy router, do którego dociera datagram. Datagram z wartością TTL równą 0 usuwany jest z sieci, co zapobiega krążeniu w niej błędnie zaadresowanych datagramów.

Pole *protokół* określa typ protokołu będącego nadawcą datagramu i jednocześnie protokół który powinien odebrać datagram w punkcie docelowym.

Kluczowe znaczenie dla wyznaczania trasy przesyłania datagramu mają pola *adres docelowy* i *adres źródłowy*. Na podstawie wartości pola Adres docelowy protokół IP dynamicznie wyznacza kolejny węzeł sieci na drodze pakietu, proces ten nazywamy marszrutowaniem IP. Adres IP jest 32–bitowym numerem, który w sposób unikalny wyznacza miejsce podłączenia interfejsu sieciowego (karty sieciowej) do sieci. W adresie IP wyróżniamy część globalną określającą adres sieci i część lokalną identyfikującą komputer w konkretnej sieci. Istnieje pięć klas adresów IP (Rys.: 2.4):

Aby zapewnić unikalność adresów sieci przydzielane są one centralnie przez organizację InterNIC. Adres zapisuje się zwykle w tzw. notacji kropkowo – dziesiętnej, dzieli się go



Rysunek 2.4: Klasy adresów IP

na cztery ośmiobitowe grupy i wartość każdej z nich zapisuje dziesiętnie np.156.17.130.10 jest adresem serwera pocztowego Instytutu Sterowania i Techniki Systemów Politechniki Wrocławskiej.

Istnieją trzy typy adresów IP: adres unicast dotyczy pojedynczego komputera, adres broadcast dotyczy wszystkich komputerów w danej sieci a adres multicast obejmuje grupę komputerów należących do jednej grupy multicastowej. Obecnie wszystkie implementacje IP muszą obsługiwać tzw. adresowanie podsieci. Zamiast traktować adres IP jako proste złożenie adresu sieci i hosta (komputera) w części adresu wskazującej hosta wydziela się adres podsieci i właściwy adres hosta. Wynika to z faktu, że adresy klasy A i B przeznaczają zbyt wiele bitów na adres hosta, adres klasy B umożliwia np. zaadresowanie 65535 komputerów, a rzadko spotyka się tak wiele hostów podłączonych do jednej sieci. Podział 16 bitowego adresu hosta klasy B na dwie części ośmiobitowe umożliwia utworzenie w jednej sieci 255 podsieci z 254 hostami w każdej z nich. Decyzję o podziale na podsieci podejmuje lokalny administrator, a informację o tym jaka ilość bitów przeznaczona jest na adres podsieci przechowywana jest przez tzw. maskę podsieci. Jest to wartość 32–bitowa zawierająca jedynki dla części adresującej sieć i podsieć oraz zera dla części adresującej hosta. Maska podsieci ma w omówionym przypadku postać 255.255.255.224.

Istnieje kilka zarezerwowanych adresów IP wykorzystywanych do celów specjalnych. Obowiązkowym adresem specjalnym jest adres interfejsu pętli zwrotnej (ang. *loopback*). Większość systemów przypisuje temu interfejsowi z definicji adres 127.0.0.1, choć poprawny jest dowolny adres klasy A z adresem sieci 127. Datagramy kierowane na adres pętli zwrotnej przechodzą przez warstwę transportową i sieciową po czym są zawracane w górę stosu TCP/IP. Najczęstszym zastosowaniem pętli zwrotnej jest testowanie poprawności działania protokołów TCP/IP. Niektóre systemy (np. system AIX) umożliwiają dodawanie aliasów (dodatkowych adresów) do adresu interfejsu pętli zwrotnej.

Każdy komputer (host) podłączony do sieci IP posiada co najmniej jeden adres IP, hosty posiadające kilka interfejsów sieciowych posiadają po jednym adresie dla każdego interfejsu. Wszystkie te adresy, wraz z adresami specjalnymi i adresami typu broadcast przechowywane są w tzw. tablicy marszrutowania (routingu). Protokół IP po otrzymaniu datagramu sprawdza, czy jego adres IP zgodny jest z którymś z adresów hosta lub jednym z adresów specjalnych. Jeśli tak jest datagram zostaje zaakceptowany. Postępowanie w przeciwnym wypadku zależy od tego, czy warstwa IP skonfigurowana jest jako router.

Jeżeli host nie jest routerem następuje odrzucenie datagramu, w przeciwnym wypadku uruchamiany jest opisany poniżej algorytm marszrutowania.

Marszrutowanie IP dokonywane jest na podstawie kolejnych przejść pomiędzy węzłami sieci zapisanych (wraz z adresami własnymi hosta) w tablicy marszrutowania. Pojedynczy rekord tablicy routingu zawiera znany adres docelowy IP i adres IP na który należy skierować datagram jeśli jego adres docelowy jest zgodny z adresem zapisanym w rekordzie. Jeżeli nadawca i odbiorca są bezpośrednio połączeni rekord tablicy marszrutowania zawiera wskazanie (w postaci adresu IP) na adresata datagramu, o czym informuje specjalny znacznik w danym rekordzie. W przeciwnym wypadku jest to adres routera stanowiącego połączenie z inną siecią lokalną lub punkt wyjścia do sieci rozległej. Zakłada się że router taki znajduje się bliżej punktu docelowego datagramu, gdyż IP nie zna pełnej trasy dla żadnego z datagramów adresowanych poza sieć lokalną. Wybór drogi datagramu przebiega następująco [**?**]: w pierwszej kolejności poszukiwany jest rekord zawierający w pełni zgodny adres docelowy IP. Jeśli zostanie on znaleziony, datagram jest wysyłany na adres wskazany przez rekord. Może być to adres docelowy lub znane przejście przez router do innej sieci lokalnej. Następnie w tablicy marszrutowania poszukiwany jest rekord z adresem sieci zgodnym z adresem sieci marszroutowanego datagramu. Jeśli poszukiwania zakończą się sukcesem datagram jest wysyłany na zawarty tam adres. W ten sposób mogą być obsługiwane datagramy zaadresowane do hostów połączonych z nadawcą przez wspólną sieć (w sensie zgodności adresów globalnych IP). W przypadku niepowodzenia poprzednich poszukiwań datagram wysyłany jest na adres IP routera domyślnego.

Ostatnią czynnością dokonywaną zanim datagram rzeczywiście opuści komputer nadawcy jest ustalenie jego adresu fizycznego w danej sieci lokalnej (np. adresu Ethernet lub Token Ring). Każde urządzenie: komputer, router, czy most (bridge) posiada unikalny w obszarze danej sieci lokalnej adres fizyczny. Odwzorowanie adresu IP w adres fizyczny dokonuje się przy użyciu protokołu ARP (ang. *Address Resolution Protocol*), opis jego działania leży poza zakresem tej pracy. Wypada zaznaczyć, że adresy IP klasy B wyczerpały się w roku 1997, co wymusiło wprowadzenie nowej specyfikacji protokołu IP tzw. IPv6, który wprowadza adresy 128 bitowe.

2.4 Protokół TCP

Protokół TCP (ang. *Transmission Control Protocol*) zapewnia niezawodne, zorientowane połączeniowo, oparte na strumieniach bajtów (tzn. nie stosujące żadnego arbitralnego podziału przesyłanych danych na jednostki stałej długości) usługi warstwy transportowej. Przed rozpoczęciem transmisji dwie aplikacje używające TCP muszą ustanowić połączenie w postaci wirtualnego kanału transmisyjnego. Niezawodność tego połączenia zapewniana jest przez działania takie jak: podział przesyłanych danych na segmenty o rozmiarze optymalnym ze względu na poprawność transmisji; utrzymywanie dla każdego segmentu osobnego zestawu zegarów wyznaczających m. in. maksymalny czas oczekiwania na po-

twierdzenie odbioru segmentu i wymuszającego retransmisję w przypadku przekroczenia tego czasu; używanie obowiązkowych sum kontrolnych; odrzucanie zdublowanych datagramów IP; zdolność rozpoznania kolejności datagramów IP (mogą one docierać w kolejności innej niż były wysyłane).

Informację kontrolną segmentu TCP zawiera przedstawiony poniżej nagłówek [**?**].

<table>
<tr><td colspan="8">Numer portu źródłowego (16 bitów)</td><td colspan="8">Numer portu docelowego (16 bitów)</td></tr>
<tr><td colspan="16">Numer sekwencyjny (32 bity)</td></tr>
<tr><td colspan="16">Numer potwierdzenia (32 bity)</td></tr>
<tr><td colspan="4">Długość nagłówka (4 bity)</td><td colspan="4">Zarezerwowane (6 bitów)</td><td>URG</td><td>ACK</td><td>PSH</td><td>RST</td><td>SYN</td><td>FIN</td><td colspan="4">Rozmiar okna (16 bitów)</td></tr>
<tr><td colspan="8">Suma kontrolna segmentu TCP (16 bitów)</td><td colspan="8">Wskaźnik ważności (16 bitów)</td></tr>
<tr><td colspan="16">Opcje</td></tr>
<tr><td colspan="16">Dane</td></tr>
</table>

Rysunek 2.5: Format segmentu TCP z wyróżnieniem nagłówka.

Pola *numer portu źródłowego* i *numer portu docelowego* zawierają wielkości całkowite i pełnią kluczową rolę w identyfikacji połączeń TCP. Numery portów od 1 do 1023 są zarezerwowane dla tzw. usług dobrze znanych i zazwyczaj przydzielane są aplikacjom pracującym jako serwery. Przykładowo serwery WWW (protokół HTTP) otrzymują numer portu 80, serwery FTP wykorzystują porty 20 i 21 a serwery pocztowe (SMTP) port 25. Do nawiązania połączenia proces klienta (np. przeglądarka WWW) uzyskuje od protokołu TCP tzw. numer krótkotrwały unikalny dla każdej aplikacji używającej TCP w danym hoście. W przypadku przeglądarki WWW może być to numer 80 lub dowolny numer z zakresu 1024... 5000. Para port źródłowy – port docelowy jest wystarczająca do identyfikacji połączenia po stronie klienta. Jeśli nawet użytkownik uruchomi dwie identyczne aplikacje (np. dwie przeglądarki WWW), to otrzymają one różne numery krótkotrwałe. Po stronie serwera sytuacja jest trudniejsza, typowo serwer poprzez multipleksację portu obsługuje wiele połączeń z użyciem jednego numeru portu źródłowego (np. 80 dla serwera WWW), może się więc zdarzyć, że dwóch niezależnych klientów zgłosi ten sam numer krótkotrwały portu docelowego. Do jednoznacznej identyfikacji połączeń wykorzystuje się kombinację numeru portu i adresu IP nazywaną gniazdem. Gniazda są jednym z podstawowych pojęć interfejsu programisty TCP/IP.

Jak już wspomniano TCP przesyła dane w postaci strumienia bajtów, w praktyce oznacza to, że wysyłany segment danych może mieć każdorazowo inną długość. Do umożliwienia uformowania z segmentów oryginalnego pakietu służy wartość pola *numer sekwencyjny*. Jeśli rozpatrzymy strumień bajtów przesyłany tylko w jedną stronę, to wartość tego pola określa numer jaki ma w strumieniu ostatni bajt segmentu. Wartość początkową numeru sekwencyjnego ustala nadawca w momencie ustanawiania połączenia. Jeśli początkowym numerem sekwencyjnym było 100, a pierwszy segment ma długość 1024 bajty, to jego numerem sekwencyjnym jest 1123. Pierwszy bajt kolejnego segmentu składającego

się na pakiet będzie miał numer 1124, jeśli segment ten ma długość 125 bajtów, to jego numerem sekwencyjnym jest 1248. Koniec strumienia oznacza się przez ustawienie flagi bitowej FIN w nagłówku ostatniego segmentu.

TCP wymaga potwierdzeń odbioru każdego przesłanego segmentu. Służy do tego pole *numer potwierdzenia*. Jeśli znów rozpatrywać przesyłanie danych w jedną tylko stronę, to po odebraniu segmentu o numerze sekwencyjnym 1123 klient wysyła (pozbawiony danych) segment TCP z ustawioną flagą bitową ACK i wartością numeru potwierdzenia 1124 (jest to numer kolejny oczekiwanego bajtu). Aby zapewnić niezależne przesyłanie danych w obydwu kierunkach, każda ze stron pamięta numery sekwencyjne wymienianych segmentów. Opisany powyżej segment potwierdzenia posiada więc ustalany przez jego nadawcę numer sekwencyjny (np. 136), który serwer musi potwierdzić ustawiając flagę ACK i umieszczając w polu Numer sekwencyjny odpowiednią wartość (tutaj 137). Konsekwencją takiej strategii wymiany potwierdzeń jest fakt, że w trakcie wymiany danych flaga ACK jest ciągle ustawiona (ma wartość logiczną 1). Pole Długość nagłówka zawiera liczbę 32–bitowych słów składających się na nagłówek, następuje po nim 6 zarezerwowanych bitów i kolejno 6 flag bitowych. Istotną flagą jest SYN, która jest ustawiona podczas wymiany segmentów nawiązujących połączenie i ustalania początkowych numerów sekwencyjnych.

Pole *rozmiar okna* zawiera wartość całkowitą, przy pomocy której odbiorca deklaruje jaką ilość bajtów nadawca może wysłać bez oczekiwania na potwierdzenie (inaczej – jaką ilość bajtów odbiorca jest w stanie przyjąć jednorazowo). Pole ma zastosowanie w przypadku transmisji wg tzw. protokołu przesuwnego okna, a jego wartość może się zmieniać w trakcie transmisji.

Ważną rolę w trakcie nawiązywania połączenia pełni pole *opcje*. W polu tym uczestnicy połączenia wymieniają się informacją jak duży jest pojedynczy segment, który są w stanie przyjąć. Ustalone w ten sposób maksymalne wielkości segmentu MSS (ang. *Maximum Segment Size*) pozostają stałe dla obydwu końcówek połączenia przez cały czas jego trwania.

Wartość tę ustala się z uwzględnieniem nakładanego przez warstwę fizyczną ograniczenia na maksymalny rozmiar datagramu IP (MTU).

Aby móc zarządzać połączeniami oprogramowanie protokołu TCP utrzymuje tablicę połączeń. Pojedynczy rekord takiej tablicy opisuje jedno połączenie i zawiera pola:

- Status (zamknięte, w trakcie zamykania, oczekiwanie na potwierdzenie itp.)
- Adres lokalny – adres IP hosta źródłowego (przechowującego tablicę)
- Port lokalny – numer portu hosta źródłowego
- Adres zdalny – adres IP hosta docelowego
- Port zdalny – numer portu docelowego

Protokół TCP wykorzystywany jest przez wiele protokołów wyższych warstw, dla których najistotniejszą sprawą jest zapewnienie niezawodności transmisji. TCP używają m. in. HTTP, FTP czy SMTP, które zostaną krótko omówione w dalszej części tej pracy. System DNS korzysta z protokołu TCP podczas wymiany danych pomiędzy serwerami DNS w trakcie uaktualniania ich tablic odwzorowań.

2.5 Protokół UDP

Protokół UDP (ang. *User Datagram Protocol*) jest prostym, bezpołączeniowym protokołem warstwy transportowej wykorzystującym datagramy. Z każdego pakietu danych skierowanego do UDP przez warstwy górne formowany jest jeden datagram UDP, z którego (inaczej niż w przypadku TCP) tworzony jest dokładnie jeden datagram IP. Aplikacja wysyłająca pakiet musi sama zadbać o to, by jego długość nie przekroczyła MTU. UDP (w przeciwieństwie do TCP) nie implementuje żadnych mechanizmów zapewniających niezawodność transmisji. Nagłówek datagramu UDP ma bardzo prosty format [**?**].

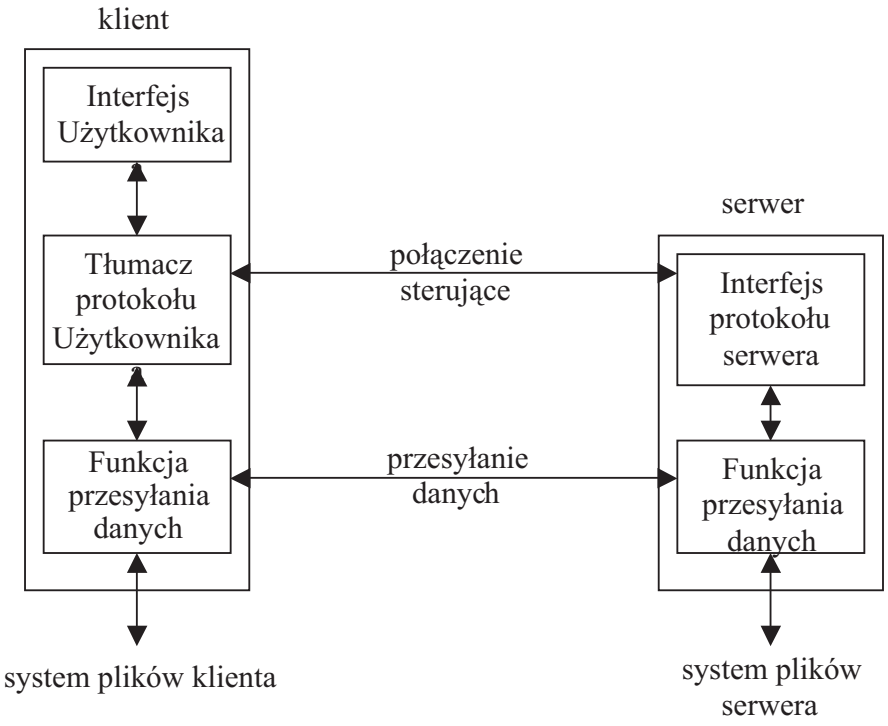
Numer portu źródłowego (16 bitów)	Numer portu docelowego (16 bitów)
Długość datagramu UDP w bajtach (16 bitów)	Suma kontrolna datagramu UDP (16 bitów)
Dane (0-65527 bajtów)	

Rysunek 2.6: Datagram UDP z wyróżnionym nagłówkiem.

2.6 Protokół FTP

Protokół FTP (ang. *File Transfer Protocol*) jest standardowym w Internecie protokołem przesyłania dowolnego rodzaju plików. Jest to protokół warstw wyższych, i przez to jest często błędnie utożsamiany z aplikacjami klienta FTP. FTP korzysta w warstwie transportowej z usług protokołu TCP. Do obsługi transmisji FTP, w przeciwieństwie do większości protokołów, otwiera aż dwa połączenia TCP : połączenie sterujące o numerze portu docelowego 21 i połączenie transmisji danych o numerze portu 20. Przez połączenie sterujące aplikacje FTP wymieniają zdefiniowane w protokole polecenia i potwierdzenia. Połączenie transmisji używane jest do binarnego przesyłania danych. Schemat obsługi połączenia przez procesy klienta i serwera FTP przedstawia rysunek 2.7.

Połączenie przesyłania danych nawiązywane jest dla każdego przesyłanego pliku. FTP może przesyłać pliki w jednym z trzech trybów: jako strumień bajtów, jako serię bloków podzielonych nagłówkami lub w rzadko stosowanym trybie z kompresją FTP (kompresji podlegają tylko ciągi zerowych bajtów). FTP rozpoznaje formaty ASCII i EBCDIC –



Rysunek 2.7: Obsługa połączenia ftp.

przesyłane są one w postaci rekordów o stałej długości, pozostałe pliki traktowane są jako pliki binarne i przesyłane jako strumień bajtów.

Polecenia FTP wymieniane przez połączenie sterujące mają postać ciągów wielkich liter ASCII o długości 3 lub 4 bajtów. Każde nawiązanie połączenia z serwerem FTP wymaga podania nazwy użytkownika i hasła, jest to prosty mechanizm zapewniający kontrolę dostępu do zasobów serwera. O tym, jakie pliki udostępniać konkretnym użytkownikom decyduje administrator serwera FTP. Według badań prowadzonych w szkieletowej sieci NSFNET dane FTP stale stanowią ok. 20% pakietów transmitowanych w tej sieci.

2.7 Protokół SNMP

Protokół SNMP (ang. *Simple Network Management Protocol*) pierwotnie zaprojektowano jako uniwersalne narzędzie, które umożliwi zdalne nadzorowanie śluz (ang. *gateway*) i routerów w sieciach rozległych. W miarę jego rozwoju protokół uzupełniono o możliwość zarządzania wszelkim typem urządzeń sieciowych w sieciach TCP/IP. SNMP każdej sieci opiera się o trzy składniki:

- MIB (ang. *Management Information Base*) jest to baza danych opisująca stan danego urządzenia;

- SMI (ang. *Structure and Identification of Management Information*) jest zestawem powszechnie stosowanych schematów służących odwoływaniu się do zmiennych MIB;
- SNMP określa zasady komunikowania się pomiędzy aplikacją zarządzającą siecią (serwerem SNMP), a procesem sterującym urządzeniem (agentem).

Każdy element sieci, który ma być zarządzany przez SNMP, musi przechowywać informację o parametrach określających stan danego elementu, a mających wpływ na pracę sieci. Baza danych zawierająca te parametry to właśnie MIB. W przypadku urządzenia takiego jak router rekord MIB może np. zawierać informację o stopniu zapełnienia jego buforów pamięci. MIB nie musi dotyczyć elementów ściśle sprzętowych istnieją np. MIB systemów operacyjnych. Sposób odwoływania się do elementów MIB określają schematy SMI.

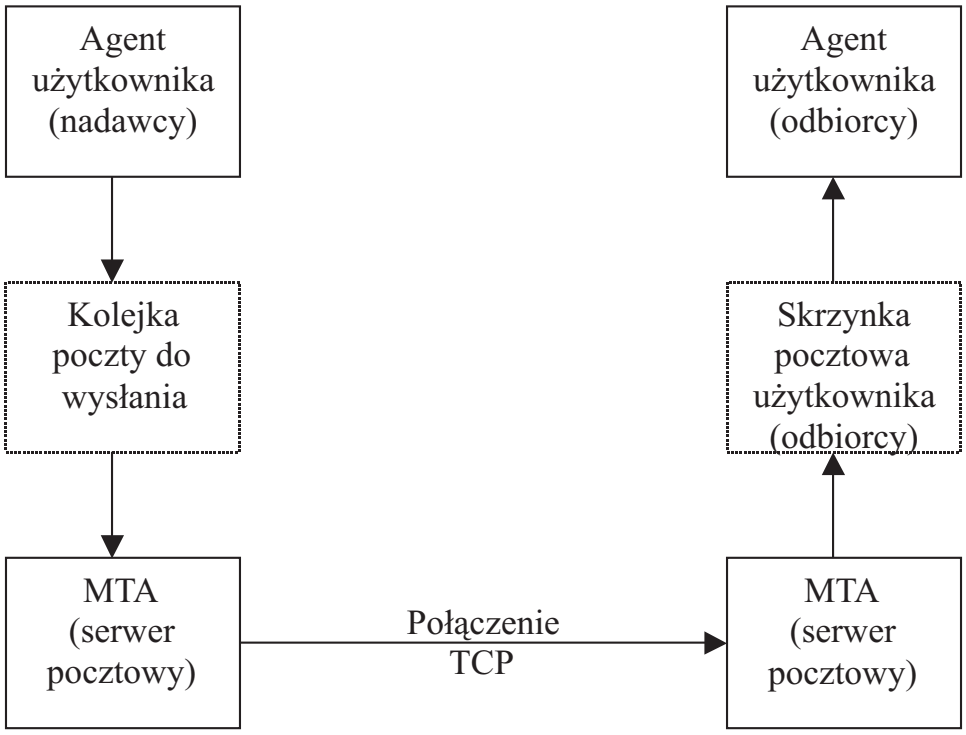
W każdej sieci, której elementy zarządzane są według standardów SNMP istnieje wydzielony host, na którym pracuje specjalistyczna aplikacja (serwer SNMP) służąca monitorowaniu i zarządzaniu pracą sieci. Proces, który działa w zarządzanym elemencie i aktualizuje MIB oraz udostępnia serwerowi jej zmienne lub ustawia je na zadane przez serwer wartości nazywany jest agentem SNMP.

W celu umożliwienia komunikacji pomiędzy serwerem a agentami SNMP definiuje pięć typów komunikatów. Trzy z nich wysyłane są zawsze w kierunku serwer–klient i służą pobieraniu lub ustawianiu zmiennych MIB. Dwa typy komunikatów wysyłane są w kierunku klient–serwer. Jeden z nich zwraca żądaną lub nowo ustawioną wartość zmiennej MIB, drugi jest komunikatem alarmowym wysyłanym przez agenta w przypadku błędnej pracy urządzenia lub przekroczenia krytycznej wartości przez którąś ze zmiennych MIB. Serwer SNMP musi utrzymywać w miarę aktualną informację o stanie sieci, dlatego agenci SNMP przepytywani są w regularnych odstępach czasu, posiadając jednocześnie możliwość raportowania o sytuacjach awaryjnych. Fakt, że cztery z pięciu możliwych rodzajów komunikacji pomiędzy serwerem a agentem SNMP to pary typu pojedyncze pytanie–pojedyncza odpowiedź (wyjątek stanowią komunikaty alarmowe) oraz chęć minimalizacji obciążenia sieci powodowanego przez pakiety służące jej zarządzaniu spowodowały, że SNMP w warstwie transportowej korzysta z usług UDP. Zawodność tego protokołu sprawia, że serwery SNMP powinny (szczególnie w sytuacjach awaryjnych) stosować mechanizmy odmierzania czasu na odpowiedź agenta i ewentualnej retransmisji komunikatów. Warto dodać, że SNMP nie jest zależny w warstwie sieciowej od IP, przy odpowiedniej konfiguracji rolę protokołu sieciowego może spełniać np. IPX/SPX.

2.8 Protokół SMTP

Protokół SMTP (ang. *Simple Mail Transfer Protocol*) jest prostym protokołem warstw górnych wykorzystywanym do przesyłania poczty elektronicznej w sieciach TCP/IP. W

warstwie transportowej SMTP wykorzystuje protokół TCP i posiada przydzielony numer portu 25. Należy wyjaśnić, że programy pocztowe, z którymi kontaktują się szeregowi użytkownicy Internetu stanowią jedynie końcowe narzędzia, w terminologii fachowej nazywane agentami użytkownika UA (ang. *User Agent*) służące do pobierania listów z elektronicznych skrzynek pocztowych lub umieszczania ich w kolejce do najbliższego serwera pocztowego. Programy rzeczywiście zajmujące się przesyłaniem wiadomości pomiędzy serwerami pocztowymi nazywane są agentami przesyłania komunikatów MTA (ang. *Message Transfer Agent*). Przykładem MTA jest unixowy Sendmail. Protokół SMTP służy właśnie do komunikacji pomiędzy poszczególnymi MTA. Schemat przesyłania poczty w Internecie przedstawia poniższy rysunek.



Rysunek 2.8: Schemat przesyłania poczty elektronicznej

Komunikacja pomiędzy MTA odbywa się według modelu klient – serwer, przy czym klientem nazywamy MTA, który inicjuje dane połączenie. Połączenie rozpoczyna się od przesłania danych identyfikujących nadawcę, a w odpowiedzi nadchodzi pakiet identyfikujący odbiorcę. Następnie przesyłana jest właściwa wiadomość i połączenie jest zamykane. Do obsługi poczty elektronicznej SMTP definiuje tylko 13 poleceń (dla porównania FTP posiada 40 poleceń), które są czterobajtowymi ciągami wielkich liter ASCII.

Wiadomość elektroniczna składa się w SMTP z trzech części. Koperty czyli danych pozwalających MTA zidentyfikować nadawcę i odbiorcę wiadomości, nagłówków czyli

danych wykorzystywanych przez agentów użytkownika (np. data wysłania wiadomości) i właściwej wiadomości. Pakiety SMTP stanowią od 8 do 13 procent pakietów transmitowanych w sieci szkieletowej NSFNET.

2.9 Protokół POP3.

Opisany powyżej protokół SMTP jest efektywnym narzędziem umożliwiającym przesyłanie poczty elektronicznej pomiędzy pracującymi w sposób ciągły serwerami SMTP. Aby pojedynczy użytkownik sieci mógł odbierać pocztę bezpośrednio przez SMTP w jego stacji roboczej musiałby ciągle działać proces typu MTA, co jest często niemożliwe ze względów czysto technicznych. Rozwiązanie tego problemu jest bardzo proste. W podłączonych do Internetu sieciach lokalnych wydziela się działający non–stop serwer pocztowy wraz z oprogramowaniem zachowującym na dysku wiadomości nadchodzące do użytkowników sieci. Protokół POP3 (ang. *Post Office Protocol version 3*) umożliwia dynamiczny dostęp do wiadomości przechowywanych na takim wydzielonym serwerze [**?**]. Protokół ten korzysta z TCP i numeru portu 110. POP3 definiuje 13 poleceń w postaci czterobajtowych ciągów ASCII.

Połączenie POP3 rozpoczyna się od identyfikacji użytkownika i podania hasła następnie serwer i klient wymieniają dane po czym sesja kończy się.

Producenci aplikacji pocztowych (UA) stosują dwa protokoły, SMTP do bezpośredniego wysyłania poczty i POP3 do odbierania jej z serwera pocztowego.

Pakiety POP3 najczęściej spotkać można w sieciach lokalnych, gdzie mogą stanowić znaczny procent transmitowanych danych.

2.10 Protokół SSL

Protokół SSL (ang. *Secure Socket Layer*) jest protokołem rezydującym bezpośrednio nad protokołem TCP i dostarczającym dowolnemu protokołowi wyższych warstw (np. HTTP) przezroczyste usługi mające zapewnić poufność przesyłanych danych. SSL wykorzystuje port TCP o numerze 443. Bezpieczeństwo połączenia oparte jest o trzy cechy protokołu SSL: symetryczne szyfrowanie danych w oparciu o np. algorytmy DES lub RC4; autoryzację hostów z wykorzystaniem szyfrowania asymetrycznego lub z kluczem publicznym (RSA, DSS) i kontrolę poprawności transmisji poprzez zastosowanie szyfrowanych sum kontrolnych MAC (ang. *Message Autenthication Codes*).

W protokole SSL wyróżniamy dwie warstwy: warstwę powitania i warstwę rekordów. Warstwa powitania (ang. *Handshake Layer*) inicjuje połączenie pomiędzy klientem a serwerem. Po wzajemnej identyfikacji hostów następuje faza negocjacji algorytmów, które będą użyte do szyfrowania transmisji (wybór ten zależy od mocy obliczeniowej hostów). Od momentu ustalenia sposobu kodowania wszystkie dane są szyfrowane.

Warstwa rekordów (ang. *Record Layer*) rezyduje pomiędzy warstwą powitania a TCP i jest odpowiedzialna za enkapsulację danych napływających z wyższych warstw do postaci rekordów o maksymalnej długości 16384 bajtów, szyfrowanie rekordów oraz dołączanie MAC [**?**].

Ponieważ SSL nie interpretuje napływających do niego danych istnieje możliwość przesyłania hipertekstu z użyciem SSL. Tak skonfigurowany serwer WWW nazywamy serwerem HTTPS lub serwerem SSL. Szyfrowane są wówczas wszystkie elementy transmisji tzn. zarówno dokumenty jak i same polecenia HTTP, co wymaga odpowiedniej konfiguracji przeglądarki.

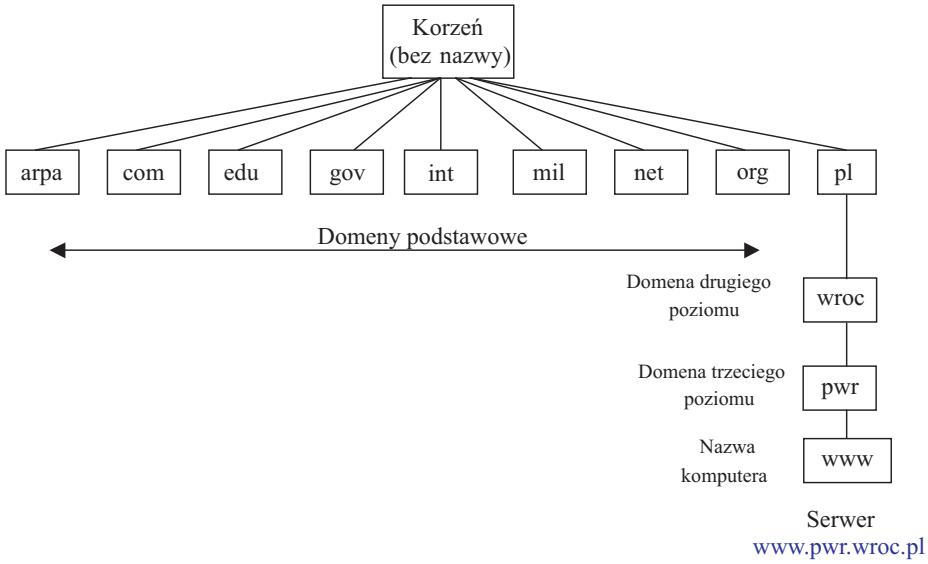
2.11 URL i DNS

Użytkownik Intrenetu do identyfikacji zasobów sieciowych stosuje zazwyczaj adres URL (ang. *Uniform Resource Locator*). URL zapisujemy w postaci [**?**, **?**]:
typ_usługi://nazwa_serwera.nazwa_domeny/ścieżka_dostępu/nazwa_zasobu. Na przykład URL http://www.netscape.com/main.html jest wskazaniem na zapisany w postaci pliku HTML dokument hipertekstowy main.html znajdujący się na ścieżce dostępu (może być to nazwa katalogu lub jej alias) News/Sport w serwerze home umieszczonym w domenie netscape.com. Dokument ten dostępny jest poprzez protokół HTTP.

Część URL określająca typ usługi interpretowana jest przez aplikację używaną do połączenia z Internetem np. przeglądarkę WWW. Ścieżka dostępu i nazwa zasobu przesyłane są do serwera i interpretowane przez jego system plików. Nazwa serwera i nazwa domeny są tylko nazwami symbolicznymi i muszą być zamienione na adres IP zanim zostanie nawiązane połączenie z serwerem. Zamiana ta możliwa jest dzięki systemowi DNS.

DNS (ang. *Domain Name System*) jest rozproszoną bazą danych umieszczoną na wielu Internetowych hostach, które nazywamy serwerami DNS. Każdy komputer, który chce korzystać z DNS musi pamiętać w swojej konfiguracji adres IP najbliższego serwera DNS. Nazwa DNS może mieć długość do 63 znaków. Przestrzeń nazw DNS ma strukturę hierarchicznego, podzielonego na poziomy drzewa. Mówimy, że domena wyższego poziomu com zawiera domenę niższego poziomu netscape. Rysunek 2.9 przedstawia hierarchiczną organizację DNS [**?**].

Domeny arpa, com, edu, gov, int, mil, net, org i domeny państwowe takie jak pl nazywamy domenami podstawowymi. Obejmują one następujące komputery: arpa – sieć ARPANET; com – organizacje komercyjne; edu – instytucje edukacyjne; gov – organizacje rządowe; int – organizacje międzynarodowe; mil – armia USA; net – sieci; org – pozostałe organizacje. Organizacje rządowe z krajów innych niż USA grupowane miały być w dwuliterowych domenach państwowych np. ae – Zjednoczone Emiraty Arabskie; pl – Polska; zw – Zimbabwe. Obecnie podział ten nie jest ściśle przestrzegany, wiele organizacji komercyjnych rejestruje się w domenie podstawowej (np. empik.com), inne stosują podział w ramach domen państwowych (np. creamsoft.com.pl). spotyka się również na-



Rysunek 2.9: Hierarchiczna struktura DNS

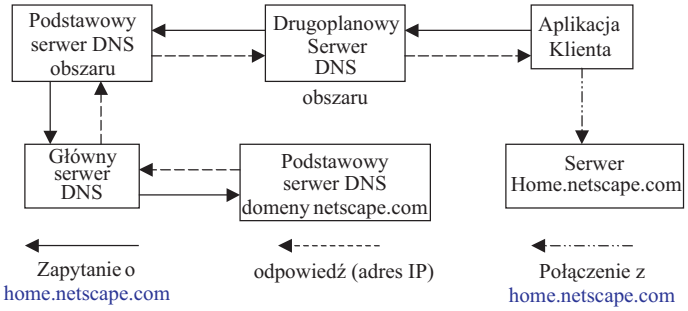
zwy w ogóle nie mieszczące się w opisanym schemacie np. www.rmf.fm.

Obszarem nazywamy oddzielnie administrowaną część drzewa DNS. Typowym obszarem jest domena drugiego lub trzeciego poziomu np. netscape.com lub pwr.wroc.pl. Jeśli w domenie zarejestrowanych jest wiele komputerów zwykle (dla zwiększenia efektywności działania DNS) dzieli się ją na kilka obszarów.

W każdym obszarze musi znajdować się podstawowy serwer DNS, który w specjalnym zestawie plików przechowuje odwzorowania wszystkich nazw komputerów z danego obszaru w ich adresy IP. Drugoplanowe serwery DNS informacje o odwzorowaniach uzyskują z serwera podstawowego korzystając z połączenia TCP o numerze portu 53. Serwery drugoplanowe przechowują (na zasadzie pamięci cache dla klientów, którzy zwracają się do nich z zapytaniami) odwzorowania obejmujące tylko część obszaru i odwzorowania o które klienci najczęściej pytają. W stałych odstępach czasu (typowo co 3 godziny) serwery drugoplanowe DNS uaktualniają swe tablice odpytując serwer podstawowy. Odwzorowanie nazwy spoza obszaru nie musi być znane ani serwerowi podstawowemu, ani drugoplanowemu, chyba że przechowywane jest w pamięci cache (jest często poszukiwane). Jeśli serwer podstawowy nie zna żądanego odwzorowania musi się zwrócić z zapytaniem (z wykorzystaniem portu 53 TCP) do jednego z serwerów głównych DNS (w 1993 roku było ich w Internecie 8), których obowiązkiem jest wskazanie serwera DNS będącego w stanie zwrócić poprawne odwzorowanie.

Proces odpowiedzialny za odwzorowanie nazw po stronie klienta nazywamy rezolwerem lub przelicznikiem nazw. Zwykle jest on zintegrowany z aplikacją (np. przeglądarką WWW) i na zasadzie pamięci cache przechowuje lokalnie pewną ilość odwzorowań. Jeśli wywoływana przez użytkownika nazwa nie jest znana lokalnie rezolwer korzystając z

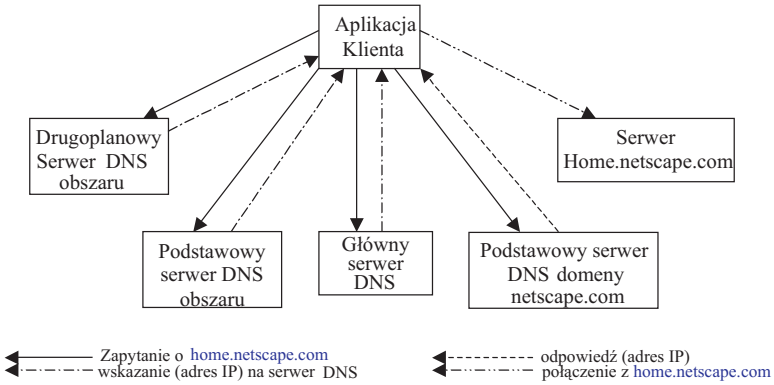
portu UDP 53 wysyła zapytanie do najbliższego drugoplanowego serwera DNS. Jeśli odpowiedź nie przekracza 512 bajtów zwracana jest w postaci datagramu UDP, w przeciwnym wypadku jako datagram UDP wysyłane jest pierwsze 512 bajtów i w nagłówku DNS ustawiana jest flaga informująca o tym fakcie. Zazwyczaj rezolwer ponawia wtedy zapytanie korzystając z portu 53 TCP, co umożliwia przesłanie pełnej odpowiedzi w jednym segmencie TCP. Rezolwer może wysłać dwa typy zapytań: zapytanie rekurencyjne umożliwia serwerowi DNS „konsultacje" z serwerami wyższego rzędu przed zwróceniem odpowiedzi; zapytanie iteracyjne wymaga od niego natychmiastowej odpowiedzi, która jeśli dany serwer nie zna odwzorowania, musi mieć postać adresu IP serwera DNS będącego (wedle „wiedzy" zapytanego serwera DNS) w stanie udzielić poprawnej odpowiedzi. Ze wskazanym serwerem DNS resolwer komunikuje się bezpośrednio. Ponieważ istnieje możliwość, że nieprawidłowe zapytanie rekurencyjne nieskończenie będzie krążyć pomiędzy serwerami DNS w nagłówku DNS zdefiniowano pole ograniczające liczbę rekurencji. Wartość tego pola zmniejszana jest o jeden przez każdy serwer DNS, który przetwarza zapytanie. Zapytanie z zerową wartością tego pola jest odrzucane i serwer komunikuje błąd. W takim wypadku rezolwer może użyć zapytania iteracyjnego, ponowić zapytanie rekurencyjne ze zwiększoną wartością pola ograniczającego rekurencje lub zgłosić użytkownikowi błąd DNS. Rysunki 2.10 i 2.11 przedstawiają obsługę obydwu typów zapytań [**?**].



Rysunek 2.10: Obsługa zapytania rekurencyjnego

Powyższe rysunki przedstawiają „najgorszy" przypadek, gdy adres IP serwera www.netscape.com znany jest dopiero przez podstawowy serwer DNS domeny (obszaru) netscape.com. W rzeczywistości istnieje duże prawdopodobieństwo, że odwzorowanie nazwy przechowywane jest przez pamięć cache jednego z bliższych klientowi serwerów. Informacje o wzajemnych odwzorowaniach nazw w adresy IP serwery DNS przechowują w postaci rekordów zasobów DNS–DNS RR (ang. *DNS Resource Record*). Rekordy te przesyłane są do aplikacji klienta jako odpowiedź na zapytanie. Pojedynczy DNS RR ma przedstawioną na rys. 2.12 strukturę [**?**].

Pole Nazwa domeny określa domenę do której odnoszą się dane rekordu. Pole Typ określa rodzaj danych np. czy rekord przechowuje odwzorowanie nazwy w adres IP, adresu w nazwę, czy tylko dodatkowe informacje o hoście. Pole Klasa ma zwykle wartość



Rysunek 2.11: Obsługa zapytania iteracyjnego

Nazwa domeny	
Typ rekordu	Klasa zapytania
Czas życia TTL	
Długość danych rekordu w bajtach	
Dane rekordu	

Rysunek 2.12: Format DNS RR

1, co oznacza internetowy adres IP (niektóre serwery DNS udostępniają również adresy należące do innych protokołów np. IPX/SPX). Pole Czas życia TTL jest liczbą sekund określającą maksymalny czas przechowywania rekordu w pamięci cache rezolwera, niestety wiele aplikacji ignoruje tę wartość.

2.12 Protokół HTTP

W tym rozdziale szczegółowo omówiona jest specyfikacja protokołu HTTP oraz te właściwości protokołu, które rzutują na wydajność WWW.

2.12.1 Specyfikacja HTTP

Protokół HTTP (ang. *HyperText Transfer Protocol*) jest prostym protokołem warstw górnych służącym do przesyłania dokumentów hipertekstowych w sieciach opartych o protokoły TCP/IP. W warstwie sieciowej HTTP korzysta z TCP i portu o numerze 80 [**?**].

Jak wspomniano dokument hipertekstowy zawierać może obok tekstu również grafikę, animacje, dźwięki oraz odsyłacze do innych dokumentów hipertekstowych, które mogą znajdować się na innych serwerach WWW, w konsekwencji często aby skompletować

cały dokument przeglądarka WWW musi otworzyć wiele połączeń z kilkoma serwerami WWW.

HTTP wykonuje tylko przesyłanie plików składających się na dokument hipertekstowy. Jest to zazwyczaj plik będący opisem dokumentu w języku HTML, który zawiera tekst, informację o rozmieszczeniu elementów dokumentu oraz pliki będące reprezentacją takich elementów dokumentu jak grafika czy dźwięk. Ponieważ HTTP nie rozróżnia typów transmitowanych plików aplikacja serwera WWW jest dość prosta w porównaniu do aplikacji przeglądarki, która musi zinterpretować treść pliku HTML i odpowiednio rozmieścić na ekranie elementy dokumentu.

Istnieją dwa rodzaje komunikatów wymienianych pomiędzy klientem a serwerem HTTP: żądania i odpowiedzi. Format żądania HTTP ma następujący format:

Linia–żądanie
Nagłówki (0 lub więcej)
<Linia pusta>
Korpus (tylko dla żądania POST)

Format linii–żądania jest natomiast taki:

Dostępne są trzy różne żądania HTTP:

- żądanie GET, które zwraca dowolną informację (dokument) określoną przez następujący po nim URL;
- żądanie HEAD, które zwraca tylko nagłówek wskazanego dokumentu, ten typ żądania wykorzystywany jest do sprawdzania odsyłaczy pod kątem aktualności lub dostępności;
- żądanie POST używane do przesyłania danych od klienta do serwera np. przesyłania zawartości formularzy wypełnianych interakcyjnie przez użytkownika. Jest to jedyne żądanie wraz z którym przesyłany jest korpus komunikatu.

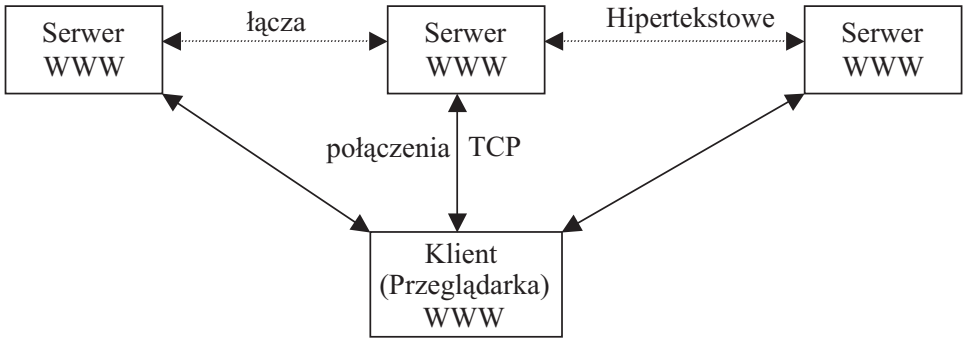
Format odpowiedzi HTTP jest następujący:

Linia–stanu
Nagłówki (0 lub więcej)
<Linia pusta>
Korpus

Format linii–stanu (status–line) ma następujący format:

Wersja–HTTP kod–odpowiedzi fraza–odpowiedzi.

HTTP definiuje 17 różnych nagłówków, które dzielimy na 3 rodzaje: nagłówki używane z żądaniami, nagłówki używane z odpowiedziami, nagłówki określające korpus (przesyłane dane). Niektóre nagłówki mogą być używane zarówno z żądaniami jak i odpowiedziami (np. nagłówek DATE). Przykładowym nagłówkiem występującym z żądaniem jest nagłówek IF-MODIFIED-SINCE, który wraz z następującym po nim nagłówkiem DATE stanowi element tzw. warunkowego żądania GET. URL występujący w takim żądaniu jest otwierany tylko w przypadku jeśli był zmodyfikowany po wymienionej w żądaniu dacie. Jednym z nagłówków używanym z odpowiedziami jest LOCATION. Jego korpus stanowi nowy URL dokumentu, którego dotyczyło poprzednie zapytanie. Przykładem nagłówka określającego korpus jest LAST-MODIFIED. Następuje po nim nagłówek DATE, który podaje datę ostatniej modyfikacji dokumentu.



Rysunek 2.13: Schemat połączeń w sieci WWW

Pierwsza linia odpowiedzi serwera nazywana jest linią stanu. Rozpoczyna ją określenie wersji HTTP, następnie podana jest trzycyfrowa liczba określająca kod odpowiedzi, a na końcu czytelne dla użytkownika wyrażenie. Poniższa tabela przedstawia znaczenie poszczególnych kodów odpowiedzi.

Starsze specyfikacje protokołu (HTTP 0.9 i HTTP 1.0) używały osobnego połączenia TCP do pobrania każdego elementu (pliku) wchodzącego w skład dokumentu. Najnowsza wersja HTTP 1.1 umożliwia stosowanie stałego, podzielonego na sesje połączenia TCP do pobrania całego dokumentu.

Poniżej omówiono istotne ze względu na wydajność WWW cechy protokołu HTTP.

2.12.2 Właściwości ruchu generowanego przez HTTP, a wydajność WWW

W okresie od stycznia 1994 do kwietnia 1995 udział komunikatów HTTP wśród wszystkich pakietów przesyłanych w szkieletowej sieci NSFNET wzrósł z 3% do 36% i wartość

ta wykazywała stałą tendencję wzrostową.

Badania prowadzone w USA wykazały niezależną od stopnia obciążenia serwera proporcję pomiędzy poszczególnymi typami żądań klienta. Żądania typu GET stanowią ok. 99% ogółu komunikatów przetwarzanych przez serwer. 0,85% komunikatów to żądania typu HEAD. Pozostałe 0,15% stanowią żądania POST.

Jeśli chodzi o odpowiedzi to 78% do 92% stanowią odpowiedzi typu 20x (odpowiedzi typu „sukces"). Odpowiedź typu 304 obejmują 4% do 14% wysyłanych przez serwer komunikatów. Łącznie odpowiedzi typu 20x i 304 stanowią 92% do 97% ogółu odpowiedzi. W przypadku pozostałych typów odpowiedzi różnice są już znaczne w zależności od rodzaju informacji na serwerze i jego popularności. Np. odpowiedzi 301 i 302 mogą mieć 0,3% udział w odpowiedziach popularnego serwera publikującego informacje naukowe NCSA (ang. *National Center for Supercomputer Applications*) do nawet 4,2% w przypadku niewielkiego serwera uniwersyteckiego w Calgary. Odpowiedzi typu 40x i 50x stanowią zwykle od 1% do 4%.

Typowo ilość danych przesyłanych w połączeniu HTTP jest niewielka. Żądania klienta nie przekraczają kilkuset bajtów, a pojedyncza odpowiedź serwera rzadko kiedy przekracza 10 kB.

Ciekawych informacji dostarczyć może analiza pomiaru czasu RTT (ang. *Round Trip Time*), czyli łącznego czasu wymiany pakietu na drodze serwer – klient – serwer. Wartość ta jest używana przez TCP do wyznaczenia czasu retransmisji segmentu w przypadku braku potwierdzenia odbioru. Przybliżenie tego czasu można uzyskać dokonując pomiaru czasu trwania fazy zamykania połączenia TCP. Przytoczone tu wyniki otrzymano mierząc ten czas dla 19 tyś. połączeń wykonanych przez 810 różnych klientów na terenie USA. Najmniejsza wartość wyniosła 0 sekund dla hosta lokalnego a największa 12,3 sekundy. Wartość średnia była równa 0,445 sekundy a mediana 0,187 sekundy. Wyniki te są zadziwiające jeśli wziąć pod uwagę, że teoretyczny RTT pomiędzy wschodnim i zachodnim wybrzeżem USA wynosi 0,06 sekundy. Odpowiedzialność za ten fakt ponosić może znaczna ilość klientów podłączonych poprzez modem (najszybszy nawet modem wprowadza ok. 0.2 sekundy opóźnienia do każdego pojedynczego pomiaru wartości RTT).

Cechą bardzo niekorzystnie wpływającą na wydajność HTTP w jego starszych wersjach była konieczność otwierania osobnego połączenia TCP dla każdego elementu dokumentu. Wiele zależało tutaj od konstrukcji przeglądarki. Jeśli nie otwierała ona połączeń równoczesnych to pobranie całego dokumentu wydłużało się o czas konieczny na kolejne nawiązywanie i zamykanie połączeń. Jeśli z drugiej strony przeglądarka „agresywnie" otwierała zbyt wiele połączeń równoczesnych, to barierą stawała się przepustowość połączenia z Internetem i wielkość MSS oferowana przez serwer. Znany jest przykład przeglądarki NCSA Mosaic, która potrafiła otworzyć kilkanaście równoczesnych połączeń TCP dla pobrania pojedynczego dokumentu, powodując tym zarówno zatory w sieci lokalnej jak i niepotrzebne obciążenie serwera.

Otwieranie zbyt wielu połączeń równoczesnych nie przynosi spodziewanych korzyści. Podstawowym problemem wydajności HTTP wydaje się być niedopasowanie protokołu

TCP – zorientowanego na przesyłanie strumienia bajtów i usługi WWW zorientowanej na przesyłanie wyodrębnionych komunikatów. Należy pamiętać że HTTP, który jest w rzeczywistości protokołem przesyłania plików, miał w swych założeniach zapewniać wydajność większą niż protokół FTP. Chciano uzyskać to eliminując występujące w FTP dodatkowe połączenie sterujące (port 21) i konieczność nawiązywania połączenia TCP na dwóch portach.

Niestety TCP wymaga zestawienia połączenia przed faktycznym rozpoczęciem transmisji. Wprowadza to opóźnienie o wartości ok. 1 RTT przed przesłaniem pliku. Dodatkowo w starszych wersjach HTTP pobranie dokumentu wymagało otwieranie nowego połączenia TCP dla każdego pliku składowego, co oznacza, że należało zamykać połączenia przez które wysłano wcześniejsze elementy dokumentu. Każdorazowe zamykanie i otwieranie połączeń wprowadza opóźnienie rzędu 3RTT dla każdego pliku składowego. HTTP 1.1 używa stałego łącza TCP dla dokumentu, co zredukowało opóźnienie przed rozpoczęciem transferu kolejnego elementu dokumentu do 1RTT. Dodatkowo HTTP 1.1 umożliwia wysyłanie żądań potokowych tzn. wysyłanie żądań o kolejne pliki przed rozpoczęciem odbierania poprzednio żądanych elementów dokumentu. Niestety w przypadku plików dynamicznych np. plików wynikowych skryptu CGI potok taki jest wstrzymywany.

Rozdział 3

Rozproszone serwery WWW

Rozdział ten przybliży architekturę najczęściej wykorzystywaną w komunikacji tak pomiędzy programami jak i urządzeniami sieciowymi: architekturę klient–serwer. Następnie zostanie przedstawiona budowa i działanie serwera WWW oraz współpracującego z nim klienta – przeglądarki. Kolejnym punktem tego rozdziału będzie klasyfikacja serwerów WWW ze względu na wielkość obsługiwanego ruchu jak i na charakterystykę budowy i wymagania oraz związane z tym definicje. W ostatnim punkcie tego rozdziału będzie przedstawiony sposób testowania serwerów webowych.

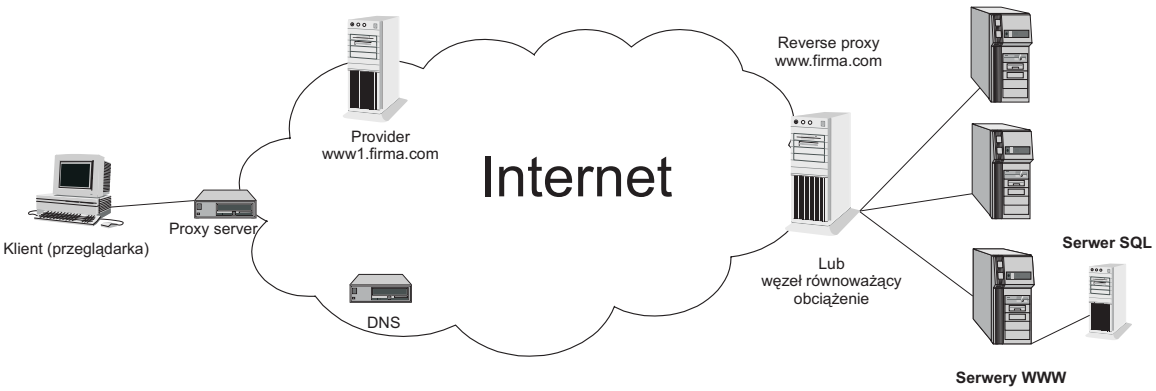
3.1 Wprowadzenie

Częstość i wydajność dostarczania usług WWW, przy stale zwiększającej się ich popularności, stanowi nie lada problem dla tradycyjnych rozwiązań klient–serwer. Zwiększenie dostępności serwisów można osiagnąć modyfikując poszczególne elementy na drodze od klienta do serwera i/lub dodając nowe. Na rys. 3.1 przedstawiono część elementów, które wpływają na wydajność sieci Web.

Najprostszymi są *cache* przeglądarki lub proxy serwer zainstalowany po stronie klienta. Takie rozwiązanie ma wiele zalet: prostota instalacji i konfiguracji, a przy odpowiednich dokumentach (statyczny HTML) efektywność takiego rozwiązania jest bardzo duża. Wadą tych rozwiązań jest głównie słaba wydajność przy dokumentach generowanych dynamicznie (a takich obecnie zdarza się coraz więcej). Podobne rozwiązanie można zastosować po stronie serwera WWW, tzn. proxy, które keszuje strony po stronie serwera(ów) – nazywa się ono *reverse proxy*.

Można także skorzystać z usług różnych dostawców (*providerów*) i przenieść część witryny WWW na ich komputery. Efektem będzie zwiększenie dostępności (oraz wydajności) witryny, jednakże za cenę mniejszego bezpieczeństwa i zmniejszenia funkcjonalności sajtu.

Można zwiększać wydajność w najprostszy sposób – poprzez dodawanie kolejnych



Rysunek 3.1: Przykładowe elementy stanowiące o wydajności WWW

procesorów, pamięci, dysków czy też urządzeń sieciowych. Jest to możliwe tylko w bardzo ograniczonym zakresie.

Wydaje się, że najciekawszym rozwiązaniem jest wielokomputerowy serwer WWW. Zaletą wykorzystania wielokomputerowego serwera WWW jest praktycznie nieograniczona skalowalność i wysoka dostepność, wadą natomiast – propagowanie ruchu na poszególne jego składowe (nody). Zarządzanie ruchem można realizować na szereg sposobów: poprzez *reverse proxy*, na poziomie serwera DNS, poprzez osobny węzeł równoważący obciążenie, modyfikując usługi po stronie klienta (opisane dalej applety Java), lub serwera (metody te zostały opisane w rozdziale 3). Wraz z architekturą takiego systemu zarządzającego serwerem WWW należy zaprojektować algorytmy umożliwiające rozpraszanie ruchu sieciowego (Rozdział 4). Część wyżej wymienionych rozwiązań może być tak iprogramowa jak i sprzętowa.

3.2 Model klient–serwer

Model współpracy klient–serwer to taki model, w którym jeden program czeka pasywnie na żądanie komunikacji wysyłane przez inne programy. Określenia klient i serwer odpowiadają dwóm programom zaangażowanym w wymianę informacji. Program inicjujący połączenie nazywany jest klientem, a program biernie czekający na żądanie połączenia – serwerem. Charakterystyka modelu [**?**]:

Oprogramowanie klienta

- dowolny program użytkowy, który staje się klientem tymczasowo (w trakcie komunikacji), ale wykonuje również obliczenia lokalnie;
- jest wywoływane bezpośrednio przez użytkownika na czas obejmujący jedną sesję;

- działa lokalnie na urządzeniu osobistym użytkownika;
- aktywnie inicjuje połączenie z serwerem;
- w razie potrzeby może komunikować się z wieloma serwerami, jednak naraz aktywnie komunikuje się tylko z jednym;
- nie wymaga specjalnego sprzętu, ani spesjalizowanego systemu operacyjnego.

Oprogramowanie serwera

- jest specjalizowanym programem, którego zadaniem jest świadczenie konkretnej usługi – może obsługiwać naraz wielu klientów;
- jest programem uruchamianym podczas startu systemu i działa przez wiele kolejnych sesji;
- działa na publicznie dostępnym komputerze;
- czeka na zgłaszanie się programów klienckich;
- pełni konkretną usługę, ale połączenia przyjmuje od dowolnych odległych klientów;
- wymaga specjalnego sprzętu i wyrafinowanego systemu operacyjnego.

3.3 Architektura WWW

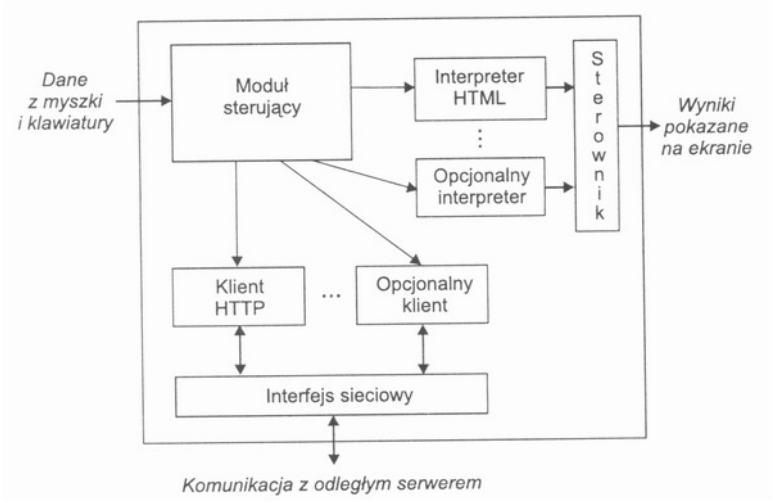
3.3.1 Serwer WWW

Najprościej opisać serwer WWW jako program wykonujący w pętli prostą operację: czekanie na otwarcie połączenia przez klienta (przeglądarkę) czyli na wysyłanie przez niego żądania dostępu do określonej strony. W odpowiedzi serwer wysyła żądany dokument (lub komunikat o błędzie w razie jego braku) albo przekazuje połączenie do realizacji modułowi (np. odpowiedzialnemu za obsługę CGI) lub innemu serwerowi (np.: baz danych). następnie serwer zamyka połączenie i czeka na następne.

3.3.2 Klient WWW – przeglądarka

Przeglądarki WWW mają bardziej złożoną budowę niż serwery WWW. Obsługuje ona większość zagadnień związanych z dostępem do dokumentów i ich pokazywaniem użytkownikowi. W związku z tym składa się ona z szeregu dużych modułów, które ze sobą współdziałają [**?**].

Koncepcyjnie przeglądarka składa się z zestawu klientów, interpreterów i modułu, który tym wszystkim zarządza. Moduł centralny – zarządzający jest odpowiedzialny za interpretację danych z klawiatury i myszki oraz za wywołania pozostałych modułów w celu wykonania operacji żądanych przez użytkownika.



Rysunek 3.2: Główne składniki przeglądarki WWW

Każda przeglądarka musi zawierać interpreter języka HTML, żeby w ogóle mogła interpretować dokumenty WWW. Inne interpretery są opcjonalne, jednakże powoli stają się również niezbędne – takie jak interpreter Javy, czy XML. Dane dla interpretera HTML stanowi dokument o zawartości zgodnej ze składnią tego języka; wynikiem jego działania jest sformatowana postać dokumentu przez zamianę znaczników formatujących na polecenia sterujące sprzętem wyświetlającym informacje.

Jedną z najważniejszych funkcji interpretera HTML jest obsługa fragmentów dokumentu wybieralnych przez użytkownika. Interpreter musi przechowywać informacje o związku między pozycjami na ekranie, a odsyłaczami w dokumencie HTML. Gdy użytkownik wskazuje za pomocą myszki pozycję w dokumencie, interpreter HTML na podstawie bieżącej pozycji kursora i zapamiętanych informacji ustala, który odsyłacz wskazał użytkownik.

3.3.3 Podział serwerów WWW

Serwery WWW można scharakteryzować głównie poprzez wielkość (ilość realizacji żądań) [**?**]:

małe – uruchamiane na stacjach roboczych, zdolne do obsłużenia do 5000 zapytań dziennie, np.: prezentujące dane jakiejś niewielkiej firmy;

średnie – uruchamiane na dużych serwerach posiadających zabezpieczenia na poziomie urządzeń wewnętrznych. Obsługują kilkadziesiąt tysięcy żądań dziennie. Prezentują dane (tysiące stron) średniej firmy;

duże – uruchamiane na serwerach o dużej mocy obliczeniowej (zwykle kilku) posiadających techniki zabezpieczenia danych i pracy serwerów. Są w stanie obsłużyć kilkaset tysięcy zapytań dziennie (prezentują kilkaset tysięcy stron);

globalne – są w stanie obsłużyć milion żądań klienckich dziennie, prawie zawsze zbudowane z wielu dużych serwerów, posiadające kilka kopii dokumentu. Są wstanie udostępniać ogromne ilości danych , również multimedialnych.

W zależności od zawartości witryn definiuje się trzy klasy obciążalności serwerów WWW [**?**]. Mamy zatem witryny typu:

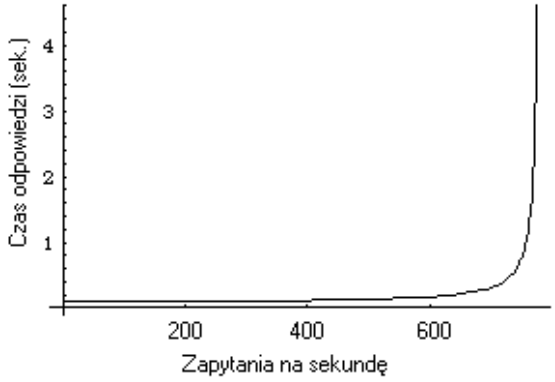
web publishing – witryny zawierające statyczne i w niewiekiej ilości dynamiczne dokumenty. Dokumenty statyczne są składowane na dyskach serwera webowego i nie podlegają modyfikacjom przez relatywnie długi czas i zawsze są przechowywane w pamięci podręcznej. Pamięć podręczna każdego nodu klastra jest zwykle ustawiona na 15% całkowitego drzewa dokumentów witryny. Strony w niewielkim stopniu dynamiczne są przechowywane w pamięci podręcznej (cache) z pradopodobieństwem 0,3.

web transaction (light) – są to witryny zawierające około 60% dokumentów statycznych i około 40% stworzonych dynamicznie dokumentów. Zapytania bazodanowe do serwerów *back–endowych* wymagają intensywanego korzystania z dysków i stąd rezultaty zapytań nie są przechowywane w keszu.

web transaction (heavy) – witryny zawierające 30% statycznych dokumentów, 30% niewiele zdynamizowanych dokumentów oraz różne kombinacje (dla pozostałych 40%) dokumentów, zwykle zawierających elementy wymagające sporych mocy obliczeniowych CPU i/lub dysku.

3.4 Charakterystyka wydajności serwera WWW

Podstawową miarą obciążenia serwera WWW jest ilość żądań HTTP, które docierają do niego w ciągu sekundy. Za miarę wydajności serwera można przyjąć widziany od strony klienta czas odpowiedzi na żądanie lub czas ukończenia transferu dokumentu, który zależny jest jednak od przepustowości połączenia klienta z Internetem. Przeprowadzone w USA symulacje z użyciem prostego modelu serwera WWW, który oparto o teorię kolejek i założenie, że istotne są tylko żądania typu GET, są podstawą do przedstawienia poniższych charakterystyk. Typowo czas odpowiedzi na żądanie rośnie wraz ze wzrostem liczby żądań na sekundę. Wzrost ten jest początkowo bardzo powolny, niemal niezauważalny aż do momentu, w którym ilość równocześnie otwartych połączeń TCP przekracza możliwości obsługi ich przez serwer. Przedstawia rysunek 3.3[26].

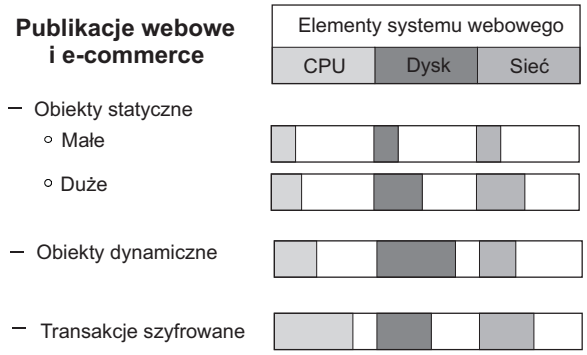


Rysunek 3.3: Czas odpowiedzi a ilość zapytań na sekundę.

Obciążenie przy którym serwer załamuje się zależy oczywiście od sprzętowych własności serwera i zastosowanego oprogramowania. Oprócz cech samego serwera najistotniejsze dla czasu odpowiedzi są przepustowość połączenia z Internetem oraz średni rozmiar plików wysyłanych przez serwer. Średni rozmiar pliku zależy w głównej mierze od rodzaju informacji publikowanych na serwerze (będzie on np. duży na serwerze oferującym wiele plików video). Pożądane jest jednak „oszczędne" projektowanie stron WWW, gdyż wzrost średniego rozmiaru pliku powoduje szybki spadek ilości zapytań na sekundę możliwych do obsłużenia. Opisane zachowanie serwera powodujące załamanie się jego wydajności po przekroczeniu maksymalnej liczby zapytań na sekundę jest typowe dla implementacji nie posiadających ograniczenia na maksymalną liczbę jednoczesnych połączeń TCP. Dotyczy to większości systemów UNIX–owych, w których procesy serwerów WWW wykonują standardową akcję fork–and–exec dla każdego nowego żądania HTTP. Prostym rozwiązaniem tego problemu wydaje się wprowadzenie ograniczenia na liczbę połączeń TCP. Wymagałoby to jednak zmian w protokole HTTP (np. wprowadzenie odpowiedzi typu „serwer zajęty – spróbuj później") oraz odpowiednich zmian w przeglądarkach.

Powyższe rozważania dotyczą wydajności serwera widzianej od strony klienta. W przypadku laboratoryjnych badań wydajności łatwiej niż czas odpowiedzi można zmierzyć ilość zapytań na sekundę obsługiwaną przez serwer. Jest to miara wydajności bezpośrednio korespondująca z czasem odpowiedzi serwera (im więcej zapytań serwer może obsłużyć, tym mniejszy jest jego czas reakcji) i wolna od wpływu takich czynników jak wydajność przyłącza po stronie klienta. Ciągły wzrost liczby użytkowników Internetu stawia coraz większe wymagania co do wydajności serwerów WWW. Do tego faktu dochodzi jeszcze jeden - charakterystyka ruchu WWW - oprócz serwowania stron statycznych, dochodzą dynamiczne i obsługa połączeń szyfrowanych. W jaki sposób zmienia się wtedy wykorzystanie poszczególnych elementów widać na rys.3.4

Prędzej czy później każdy administrator takiego serwera stanie przed problemem znale-



Rysunek 3.4: Obciążenie poszczególnych elementów serwera WWW w zależności od typu połączenia.

zienia sposobu na zwiększenie jego wydajności. Zwykle ze względu na specyfikę publikowanych informacji niewiele można zrobić ze średnim rozmiarem przesyłanego pliku. Podstawową metodą zwiększenia wydajności serwera jest zwiększenie przepustowości przyłącza. Sprzętowa modernizacja samego serwera nie przyniesie rezultatów w przypadku gdy „wąskim gardłemi" jest przyłącze. Jeśli jednak dysponujemy odpowiednio wydajnym przyłączem (np. linią ATM) „wąskim gardłem" jest na pewno serwer.

Wydajność każdego komputera zwiększyć można dokupując większą ilość pamięci RAM i szybsze dyski twarde. W przypadku serwerów WWW jest to jednak rozwiązanie prowizoryczne. Lepsze efekty przynosi instalacja maszyny wieloprocesorowej. Niestety obecne implementacje TCP/IP nie wykorzystują wielowątkowości w stopniu, który umożliwiałby pełne wykorzystanie cech technologii wieloprocesorowej. Należy się spodziewać, że otwartych połączeń TCP będzie typowo więcej niż procesorów w serwerze. Gdyby obsługę stosu TCP/IP podzielić można było na niezależne wątki, których wykonanie przebiegać by mogło z podziałem czasu, to efektywność wieloprocesorowych serwerów WWW byłaby znacznie większa niż obecnie. Współcześnie stosowanie serwerów wieloprocesorowych nie przynosi korzyści tak dużych jak spodziewane. Dobrą metodą zwiększenia wydajności WWW jest stosowanie serwerów proxy, które na zasadzie pamięci cache przechowują część dokumentów serwisu i w ten sposób częściowo wyręczają serwer główny. Najbardziej efektywną i jednocześnie najbardziej zaawansowaną technologicznie metodą zwiększenia wydajności serwera WWW jest powielenie jego zawartości na jeden lub kilka dodatkowych serwerów i udostępnienie tej grupy pod jednym adresem IP. Rozwiązanie takie wymaga sprzętu lub oprogramowania, które z jednej strony ukrywałoby przed użytkownikami istnienie kilku serwerów o jednakowej zawartości a z drugiej dokonywałoby dystrybucji obciążenia pomiędzy te serwery w sposób maksymalizujący ich łączną wydajność.

3.4.1 Serwery wielokomputerowe – rozproszone

Żeby serwery WWW były wydajne oraz mogły pracować bez przerwy przez 365 dni w roku 24 godz. na dobę – nie mogą pracować na jednym komputerze; wydajność, skalowalność a przede wszystkim odporność na awarie może zostać zrealizowana jedynie poprzez rozproszone serwery WWW, w których zapytania od klientów są w odpowiedni sposób dystrybuowane do poszczególnych serwerów [**?**].

Światowy trend w wykorzystywaniu na różne sposoby technologii World Wide Web jest coraz większy. Znakomite darmowe serwery WWW (Apache, NCSA itp.), spora ilość przeglądarek internetowych oraz olbrzymi rozwój internetu [**?**], a także ogrone możliwości tej usługi spowodowały zwiększone zainteresowanie tą technologią. Jednym z ostatnich niesłychanie modnych technologii internetowych wykorzystujących WWW jest handel elektroniczny skierowany tak do pojedynczego odbiorcy (*Business to Customer– B2C*) jak i współpracujących firm (*Business to Business*– B2B). Poza tym również bardzo interesującym celem są serwery aplikacyjne. Powoduje to jednakże konieczność utrzymania w sprawności serwisu przez cały rok – bez przerwy. Nie oznacza to wyłącznie możliwości korzystania z tych serwerów, czyli bezawaryjnej pracy, ale również uzyskanie jak najlepszego komfortu pracy z tymi systemami. Poza tym jeszcze jedną niesłychanie istotną rzeczą przemawiającą za wielokomputerowymi serwerami WWW jest ich znakomita skalowalność (dużo tańsza od skalowalności pojedynczych maszyn, które nie zawsze dadzą się rozbudowywać). Takich zalet nie posiada pojedyncza maszyna. Jej koszt zakupu, a także koszt rozbudowy – przy zapewnieniu bezpieczeństwa i odporności na uszkodzenia jest olbrzymi. Fakt, że pojedynczą maszyną łatwiej się administruje nie rekompensuje w pełni ułomności architektury von Neumanna. Wielokomputerowe serwery (nie tylko WWW, ale także serwery obliczeniowe) mogą osiągać zawrotne wydajności, są praktycznie w nieskończoność skalowalne, nadmiarowość tanich elementów (zamiast kupować trzy maszyny klasy mainframe – można kupić sto komputerów klasy PC) decyduje o ich odporności na uszkodzenia. Najlepszym przykładem jakości tej technologii jest najpotężniejsza z wyszukiwarek sieciowych – GOOGLE[1] pracująca na kilku tysiącach komputerow klasy PC, a zawierająca w swojej bazie informacje o ponad miliardzie trzystu milionach stron WWW (jest wykorzystywana przez większość serwisów webowych). Wadą systemów klastrowych jest administracja i sposób współdzielenia obciążenia pomiędzy poszczególnymi nodami (istnieje jednakże oprogramowanie darmowe umożliwiające równoważenie obciążenia serwerów WWW[2], a także serwerów obliczeniowych[3]).

Trend ten nie znajduje odzwierciedlenia tylko w serwerach webowych. Konstrukcja wielokomputerowa pozwala na znacznie efektywniejsze zarządzanie zasobami i przydzieleniem zadań do zasobów. Pozwala także uzyskać „prawdziwe" zrównoleglenie obliczeń – brak jest tu „kombinowanych" metod uwspólniania zasobów i rozdziału żądań do ma-

[1] *www.google.com*
[2] *www.linuxvirtualserver.org*
[3] *www.mosix.org*

gistral i pamięci obecnych w typowych architekturach. Kolejną zaletą systemów nieneumanowskich jest możliwość zapewnienia bezpieczeństwa ciągłości pracy w sposób dotąd niewykonalny. Przykładem spełnienia wszystkich tych cech są współczesne superkomputery i mainframy, z których doświadczeń korzystali współcześni twórcy nowoczesnych, skalowalnych technik webowych. Każdy superkomputer (SGI, Hitachi, NEC, Intel, IBM) oraz system mainframowy (głównie IBM) składa się z szeregu (nawet kilkudziesięciu) nodów skonsolidowanych zwykle w jednej obudowie, a połączonych wysokoprzepustową siecią transmisyjną. Stąd wiele pomysłów zostało zaczerpniętych i wykorzystanych przy tworzeniu rozposzonych architektur webowych.

Klasyczny system komputerowy obsługujący serwer webowy jest obsługiwany niskopoziomowo – system operacyjny obsługujący serwer webowy. W architekturze rozproszonego webu pomiędzy systemem operacyjnym poszczególnych nodów, a serwerem WWW (również poszczególnych nodów) znajduje się logiczny układ koordynujący działania poszczególnych serwerów WWW tak aby działały jako jeden rozproszony serwis. Ten system zarządzający odpowiada za:

- rozdysponowywanie żądań przychodzących od klientów na poszczególne nody w taki sposób aby zapewnić zrównoważenie obciążenia poszczególnych maszyn,
- zapewnienie spójności danych na drodze klient–serwer www–klient,
- oraz sprawowanie kontroli nad poszczególnymi elementami klastra w ten sposób, aby zawsze, i odpowiednio szybko, było wiadomo, który komputer należy omijać ze względu na awarię.

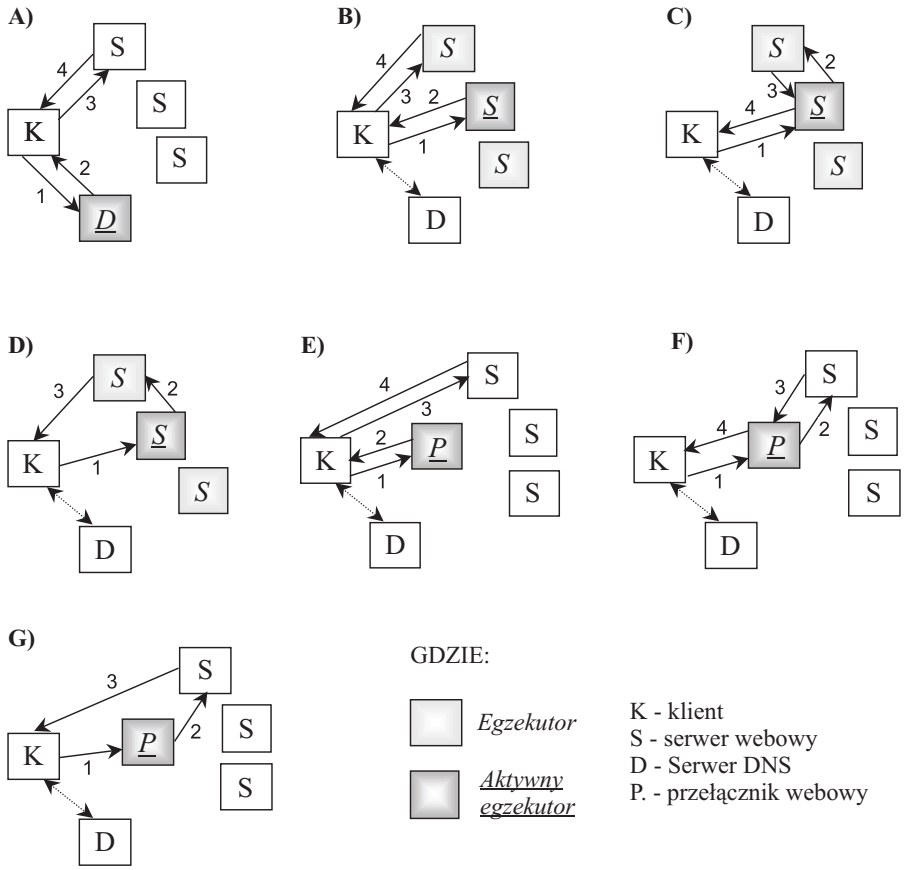
Poniżej znajduje się opis i charakterystyka poszczególnych systemów równoważących obciążenia w rozproszonych serwerach webowych.

3.5 Przegląd skalowalnych systemów serwerów webowych

Na rysunku 3.5 przedstawiono wszystkie proste modele funkcjonalne systemów do równoważenia obciążeń w serwisach WWW. Ich charakterystyczną cechą jest to, iż w każdym wykorzystywany jest jeden mechanizm szeregowania. Można je traktować jak swego rodzaju klocki, z których budowane są rzeczywiste systemy. Cześć z nich odpowiada rzeczywistym systemom bez dodatkowych modyfikacji, część odzwierciedla rzeczywiste systemy dopiero we wzajemnych kombinacjach. Wyodrębnienie prostych modeli umożliwia zbudowanie przejrzystej taksonomii skalowalnych systemów serwerów webowych.

3.5.1 Podział ze względu na rozmieszczenie serwerów webowych

Podział ten kształtuje się następująco:



Rysunek 3.5: Podstawowe modele systemów do równoważenia obciążeń

- Klaster webowy. Jeśli serwery webowe są rozproszone lokalnie (skupione geograficznie), to mówi się o tzw. klastrze webowym (web–cluster). Z technicznego punktu widzenia klaster webowy funkcjonuje w obrębie sieci lokalnej. Każdy z prostych modeli przedstawionych na rys.3.5 może być implementowany w klastrze webowym. W przypadku modelu **G**, ze względu na uwarunkowania techniczne transmisji żądań, może zajść potrzeba umiejscowienia klastra w obrębie podsieci sieci lokalnej [**?**, **?**, **?**].

- Rozproszone serwery webowe. Jeśli między rozproszonymi globalnie (rozproszonymi geograficznie) serwerami webowymi stosowany jest jakikolwiek mechanizm szeregowania, mówi się o rozproszonych serwerach webowych (distributed web–

servers), w przeciwnym przypadku – o lustrzanych serwerach webowych (mirror web–servers). Teoretycznie każdy model przedstawiony na rys.3.5 może być implementowany wśród rozproszonych serwerów webowych, jednak ze względu na powstawanie dodatkowych opóźnień podczas transmisji zwrotnej w modelu **F** oraz uwarunkowania techniczne transmisji żądań w modelu **G**, oba wymienione modele mają zastosowanie głównie klastrach webowych.

- Rozproszone klastry webowe. Szczególnym przypadkiem rozmieszczenia serwerów webowych jest kombinacja globalnego i lokalnego rozproszenia. Jeśli miedzy klastrami webowymi stosowany jest jakikolwiek mechanizm szeregowania, mówi się o rozproszonych klastrach webowych (distributed web–clusters). W przeciwnym przypadku – o lustrzanych klastrach webowych (mirror web–clusters). Rozproszone klastry webowe można przedstawiać za pomocą kombinacji prostych modeli funkcjonalnych (rys.3.6 **J, K**).

3.5.2 Podział i charakterystyka ze względu na umiejscowienie mechanizmu szeregowania

Mechanizm szeregowania może być umiejscowiony: po stronie klienta, po stronie serwera, oraz jako osobny układu (serwer DNS, dystrybutor).

Podejście od strony klienta

Główną zaletą stosowania mechanizmów rozpraszania obciążenia po stronie klienta jest całkowita niezależność od architektury serwera(ów), a przede wszystkim od miejsca ich rozmieszczenia (mogą być nie tyle geograficznie rozproszone, co wręcz nieskoordynowane jako całość).

Rozwiązania w tym zakresie możemy podzielić na dwie podgrupy:

- mechanizmy równoważenia obciążenia implementowane w przeglądarkach i jako applety Java:
  - w przeglądarkach poprzez API. Przeglądarka może aktywnie pełnić roleę w dystrybucji zapytań, pod warunkiem posiadania informacji na temat adresów serwerów WWW. Wtedy, na podstawie zapytania otrzymanego od użytkownika, przeglądarka wybiera serwer do przedłożenia zapytania. Klasycznym przykładem takiego rozwiązania jest mechanizm LB implementowany w przeglądarkach firmy Netscape [**?**, **?**]. Po wygenerowaniu zapytania przez użytkownika (tylko do serwera www.netscape.com), przeglądarka wybiera liczbę z zakresu od 1 do liczby znanych jej serwerów i kieruje zapytanie do węzła o adresie www*liczba*.netscape.com. Nie jest to rozwiązanie interesujące, gdyż liczba serwerów i adresów jest na stałe zakodowana w przeglądarce. Poza tym

taki sposób rozdysponowywania ruchu (losowy) nie zapewnia równoważenia obciążenia pomiędzy serwerami;

  - applety Java. O wiele ciekawszym od powyższego rozwiązaniem, jest implementacja specyficznych appletów Java. Gdy użytkownik wysyła żądanie do serwisu, zamiast dokumentu HTML, pobiera applet Java. Applet ten zawiera adresy IP maszyn dostarczających ten serwis WWW [**?**, **?**]. Applet może być modyfikowany przez program zbierający informacje o stanie serwerów (o ich obciążeniu) oraz o jakości połączenia sieciowego. Informacje te może wykorzystywać applet do wyboru najbardziej odpowiedniego serwera do realizacji żądania. Główną wadą tego rozwiązanie jest generowanie dodatkowego ruchu w sieci (związanej z pobieraniem informacji o stanie obciążenia serwerów. Naturalnie jest ono również bezużyteczne gdy przeglądarka nie potrafi interpretować kodu Javy.

- serwery Proxy. Jest to oparte na pomyśle po raz pierwszy wykorzystanym przez badaczy z Cambridge University [**?**]. Mechanizm, który został zaimplementowany w serwer proxy, bazuje na informacji o osiągniętej wydajności podczas dotychczasowych transmisji. Klient otrzymuje listę zreplikowanych serwerów postrzeganych przez użytkownika pod jednym adresem. Każdej pozycji na liście jest przypisana informacja o wydajności serwera. Za każdym razem, gdy klient utworzy połączenie z serwerem jest ona aktualizowana. Wybierany jest mirror na podstawie znajomości najlepszej ze średnich wydajności serwera w ciągu ostatnich transmisji. Decyzja jest podejmowana z zastrzeżeniami: jeśli wydajność ostatniej transmisji spadła poniżej pewnego progu, serwer jest uważany za nieosiągalny. Wybór serwera proxy wynikał z kilku ważnych powodów: ukrycie przed użytkownikiem faktu wyboru dokumentu między zreplikowanymi serwerami, brak konieczności ingerencji w kod przeglądarki, oraz korzyść w postaci współdzielenia informacji o wydajności zreplikowanych serwerów między użytkownikami przeglądarek. Kilku klientów generuje większą liczbę zapytań, poprawiając w ten sposób wydajność algorytmu równoważenia obciążenia [**?**].

Podejście od strony serwera

Techniki równoważące obciążenie w oparciu o serwery wykorzystują dwupoziomowy mechanizm dystrybucji. Najpierw zapytania klienta są przydzielane serwerom webowy w klastrze poprzez DNS. Następnie każdy serwer może przekazać otrzymane zapytanie do innego serwera w klastrze. Rozwiązanie w oparciu o rozproszone szeregowanie pozwala wszystkim serwerom brać udział w równoważeniu obciążenia w klastrze poprzez mechanizm przekazywania zapytań. Połączenie podejścia w oparciu o DNS oraz z technikami przekierowywania zapytań poprzez serwery webowe prowadzi do rozwiązania większości

problemów wynikających z polityki szeregowania np.: niejednolita dystrybucja zapytań wewnątrz domeny, czy ograniczona kontrola nad zapytaniami.

Propozycje oparte na przekierowaniach serwerów różnią się w sposobie podejmowania decyzji. W następnym rozdziale są przedstawione dwie główne klasy rozwiązań: wykorzystujące funkcje redirekcji na poziomie protokołu HTTP oraz w oparciu o mechanizm przepisywania pakietów.

Podejście od strony niezależnego węzła dystrybuującego zapytania

W wyniku zapytania klienta o dane z serwera WWW – w pierwszej kolejności następuje rozwinięcie nazwy domenowej serwera na jego adres IP. Następnie adres IP może być reprezentowany przez wirtualny adres IP przypisany do klastra serwerów. Samo rozwinięcie IP w adres serwera może być realizowane na różnych poziomach. Może to być przekierowanie do konkretnych zasobów na poziomie protokołu HTTP [**?**] lub na poziomie protokołu IP oraz adresu URL przy zastosowaniu dystrybutora [**?**, **?**].

Na początku skupiano się na rozwiązaniach dotyczących podmiany nazwy serwisu na jego adres IP [**?**]. Jednakże w związku z faktem ograniczeń takiego rozwiązania coraz większą uwagę poświęca się implementacji mechanizmów podmiany wirtualnego adresu IP na adres konkretnego hosta. Przykładami takiego podejścia są: Berkeley MagicRouter [**?**], CISCO Local Director [**?**], VirtualServer [**?**] oraz IBM SecureWay Network Dispatcher [**?**].

- Wykorzystanie serwera DNS – egzekutorem jest serwer DNS lub inne urządzenie wspomagające albo przejmujące rolę serwera DNS. Należy uściślić, że chodzi tu o tzw. główny serwer DNS będący autorytatywnym źródłem informacji o określonej domenie (*authoritative DNS*). Jak opisano w Rozdziale 2 system DNS służy do odwzorowania symbolicznych nazw komputerów w Internecie na ich adresy IP. Funkcja ta, w niejako naturalny sposób, czyni serwer DNS dobrym miejscem na implementację mechanizmu równoważenia obciążeń. Pierwszą instalacją wykorzystującą DNS do równoważenia obciążeń był serwis WWW NCSA (ang. *National Center for Supercomputing Applications*). Zestawiono tam klaster dziewięciu serwerów WWW, który stanowił odrębny obszar DNS i był udostępniany pod nazwą www.ncsa.ninc.edu [**?**, **?**]. Na podstawowym serwerze DNS domeny (obszaru) ncsa.ninc.edu skonfigurowano oprogramowanie, które na zapytania o odwzorowanie nazwy www.ncsa.ninc.edu odpowiadało podając cyklicznie adresy kolejnych serwerów z klastera. Ten statyczny algorytm nazywany jest cyklicznym DNS lub RR–DNS (ang. *Round–Robin DNS*).

  Podejście to jednak bardziej realizuje rozproszenie strumienia zapytań klientów niż równoważenie obciążenia serwerów WWW. W rzeczywistości w Internecie jest wiele serwerów DNA i mają one układ hierarchiczny tzn. nie zawsze trzeba sprawdzać adres IP w docelowym serwerze DNS. Oznacza to, że nie mamy wpływu na część

zapytań, jakie są kierowane do serwera. W pewnym stopniu poprawia sytuację zastosowanie sygnałów zwrotnych od serwera WWW do serwera DNS o przeciążeniu. Efekt jest odczuwalny dopiero po upływie czasu TTL (np. w momencie uszkodzenia maszyny i wyłączenia jej adresu z puli adresów serwera DNS). Aby efektywność tego typu rozwiązań była jak największa, ważne jest prawidłowe oszacowanie tzw. ukrytego obciążenia, czyli wielkości zapytań napływającego w czasie TTL. Do estymowania tej wielkości można zastosować funkcje heurystyczne [**?**].

- Reverse proxy serwer

  Typowe forward proxy (także transparent) zwykle używane przez prowajderów internetu przechowują strony najczęściej pobierane przez użytkowników. Z reverse proxy firmy mogą przechowywać specyficzną zawartość na serwerach podobnie jak prowajderzy i przekierowywać żądania (od klientów) do danych składowanych na tych serwerach poprzez proxy. Serwery proxy przechowują (keszują) informacje przychodzące od lokalnych serwerów. Zatem osiąganie stron jest o wiele szybsze – jest pobierane (o ile istnieje) z keszu serwera proxy (stąd nazwa reverse – ponieważ jego działanie jest dokładnie przeciwne do działania „zwykłego" proxy serwera).

  Z technicznego punktu widzenia reverse proxy różni się od forward proxy jednym dodatkiem – tym dodatkiem jest odpowiedni moduł tłumaczący adresy URL backendowych serwerów WWW tak jakby były to jego własne. Taki sposób działania ma jeszcze jedną ciekawą własność – serwery backendowe mogą być specjalizowane, np. niektóre z nich mogą odpowiadać wyłącznie za przetwarzanie stron dynamicznych, inne statycznych, a jeszcze inne tylko stron zawierających sporo grafik. Wystarczy w module przeadresowań umieścić odpowiednią informację (tzn. jaki adres URL w sieci wewnętrznej „tłumaczyć" na adres URL serwera proxy).

  W związku z prostotą działania takiego systemu (znakomicie wykorzystywanego jako systemu zarządzającego wielokomputerową witryną WWW – można równoważyć obciążenia, specjalizować serwery, oraz w dowolny sposób propagować ruch webowy) jest on często wykorzystywany.

  Oferta różnorakich produktów pełniących rolę reverse proxy serwerów jest bardzo bogata. Rozpościera się od produktów komercyjnych, takich jak Intel NetStructure, IBM Web Traffic Express (bedący elementem WebSphere Edge Server), po produkty niekomercyjne takie jak: Apache (najpopularniejszy serwer WWW – może być również forward, transparent oraz reverse proxy serwerem, niestety na razie obsługuje jedynie protokół HTTP do wersji 1.0), oraz bardzo wydajne narzędzie: SQUID Web Proxy Cache[4] (obsługujący wszystkie wersje protokołu HTTP, mający też niezliczoną ilość funkcji dotyczących korzystania z tego systemu i bezpieczeństwa).

[4] http://www.squid–cache.org

- Wykorzystanie dystrybutora – wyspecjalizowanego urządzenia. Alternatywnym rozwiązaniem do DNS'a, pozwalającym na pełną kontrolę ilości zapytań nadchodzących do serwerów, jest zastosowanie dystrybutora. Rozszerza to wirtualizację adresu nie tylko na poziom URL, ale również na poziom protokołu IP. Dzięki temu można zastosować jeden wirtualny adres IP (single virtual IP address) IP–SVA dla klastra serwerów. Pozwala to na pełną kontrolę nad strumieniem zapytań kierowanych do serwerów.

  Takie podejście ma jednak swoje wady. W systemie powstaje jeden węzeł obsługujący transmisję pakietów. W pewnych przypadkach wydajność całego systemu może zależeć od wydajności tego właśnie węzła. Wykorzystywane tutaj algorytmy są najprostsze, ponieważ dystrybutor obsługuje wszystkie nadchodzące pakiety i konieczne jest zminimalizowanie czasu poświęcanego na ich obsługę. Przykładem takiego rozwiązania jest SITA-V algorytm [**?**].

  Rozwiązania oparte na dystrybutorze ze względu na zastosowany mechanizm obsługi pakietów możemy podzielić na:

  – metodę packet single–rewriting; rozwiązanie to polega na przekierowywaniu pakietów nadchodzących od klientów przez dystrybutor poprzez przepisanie docelowego adresu IP. Przykładem takiego rozwiązania jest mechanizm routera TCP opisany w [**?**]. Klaster serwerów WWW składa się z kilku węzłów oraz routera TCP, który pełni rolę dystrybutora. Adres *i* jest prywatnym adresem węzła, na którym uruchomiony jest serwer WWW. Wszystkie zapytania HTTP przychodzą do dystrybutora, ponieważ tylko IP–SVA jest adresem znanym publicznie (krok 1). Wybór serwera WWW dokonywany jest przez dystrybutor na podstawie algorytmu round–robin (krok 2). Przekierowanie pakietu do odpowiedniego serwera realizowane jest dzięki przepisaniu adresu docelowego w pakiecie z IP–SVA na prywatny adres *i* serwera w klastrze. Przepisywana jest również suma kontrolna w pakiecie, ponieważ zależy ona od adresu docelowego (krok 3). W związku z tym, że jedno żądanie składa się z kilku pakietów, dystrybutor przechowuje tablicę złożoną z par: adres źródłowy – adres prywatny serwera WWW. Dzięki temu wszystkie pakiety pochodzące od jednego nadawcy mogą być kierowane do tego samego serwera WWW (krok 4). W następnym kroku serwer odsyła żądane dane do klienta (krok 6). Przed odesłaniem danych w pakiecie konieczne jest wpisanie w polu adresu nadawcy, adresu IP-SVA (krok 5).

    Jakkolwiek rozwiązanie to jest przezroczyste dla klienta, to wymaga ono dużych zmian w kodzie routera oraz systemie operacyjnym serwera. Związane jest to z tym, że następuje podmiana adresów na poziomie TCP/IP. Z drugiej jednak strony rozwiązanie to jest bardzo odporne na uszkodzenia. W przypadku awarii serwera WWW jest on usuwany z tablicy routera i nie uwzględ-

niany przy rozdziale pakietów aż do momentu naprawy. Architektura ta może być połączona z wykorzystaniem DNS'a. Pozwala to na skalowanie klastra nie tylko w sieci LAN, ale również sieci WAN.

– metodę packet double–rewriting; architektura ta w swoim założeniu jest bardzo podobna do przedstawionej powyżej. Mechanizm polega na przepisywaniu adresów docelowych w pakietach nadchodzących od klientów. Dystrybutor następnie przesyła pakiety do odpowiedniego węzła obsługującego serwer WWW. W tym przypadku jednak wszystkie pakiety wracają z powrotem do dystrybutora. Mechanizm ten opiera się na architekturze NAT (Network Address Translation) opisanej w [**?**]. W momencie odebrania nadchodzącego pakietu dystrybutor wybiera serwer WWW (krok 2), a następnie modyfikuje adres źródłowy oraz docelowy w nagłówku pakietu (krok 3). W drodze powrotnej pakietu dystrybutor ponownie zmienia adresy IP w nagłówku i przesyła dalej dane do klienta (krok 6).

  Znane są dwa rozwiązania oparte na tej architekturze. CISCO LocalDirector [**?**] oraz Magicrouter [**?**]

– metodę packet forwarding by the dispatcher. Packet forwarding jest podejściem odmiennym w stosunku do prezentowanych powyżej. Działanie jego polega na przesyłaniu pakietów do serwera w niezmienionej postaci zamiast przepisywaniu adresów w nagłówku. Podejście to pozwala na wykorzystanie tego samego rozwiązania w sieci LAN i WAN. Zarówno sama metoda jak i pakiet zostały opisane w dalszej części tej pracy.

3.5.3 Podział ze względu na strategię rozmieszczenia mechanizmów szeregowania

Z tej perspektywy rysuje się podział na dwie grupy modeli: z centralnym mechanizmem szeregowania i z rozproszonymi mechanizmami szeregowania.

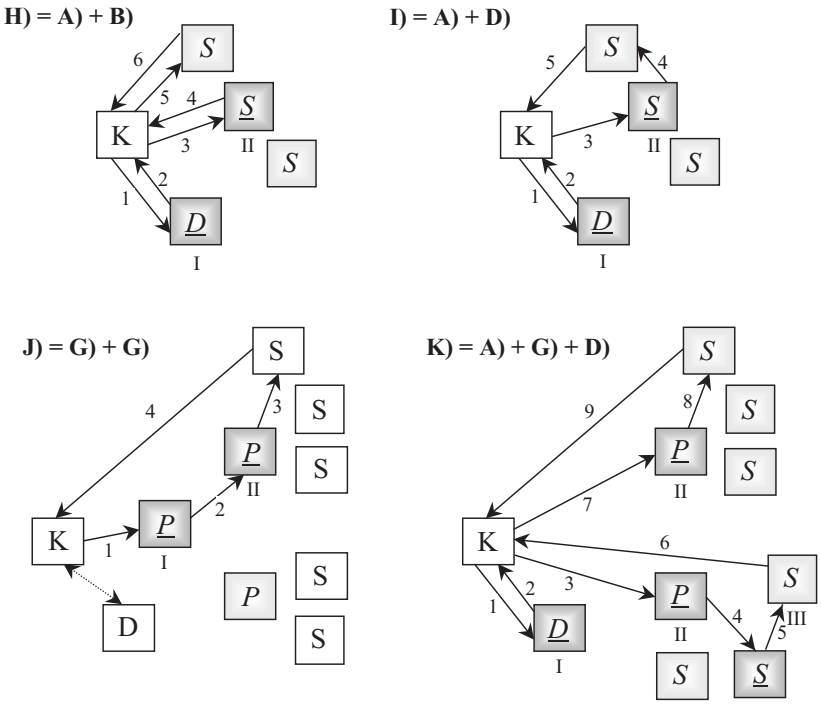
- Centralny mechanizm szeregowania. Do tej grupy należą modele **A, E, F, G** (rys.3.5). W przypadku **A** decyzja o przydzieleniu nazwie domenowej adresu IP najlepszego serwera webowego zapada centralnie, po stronie serwera DNS. Mimo scentralizowanego zarządzania, ze względu na specyfikę funkcjonowania DNS, skuteczność tego modelu jest niewielka [**?**, **?**]. W modelach decyzja o przekierowaniu żądania klienta zapada centralnie, w dedykowanym urządzeniu. W odróżnieniu od modelu, centralne zarządzanie oznacza stuprocentową kontrolę nad szeregowaniem.

- Rozproszone mechanizmy szeregowania. W tej grupie decyzja o przekierowaniu żądania może zapaść na każdym serwerze, w którym zaimplementowano oprogramowanie egzekutora. Ideę rozproszonego szeregowania ukazują modele **B, D, C**

(rys.3.5). Rozwiązanie takie zapobiega efektowi wąskiego gardła, na które w szczególności narażony jest model **F**. Modele **B** oraz **D** stosowane są głównie w celu poprawy skuteczności szeregowania w modelu **A** (rys.3.6 **H, I**).

3.5.4 Podział ze względu na liczbę stopni szeregowania

Systemy serwerów webowych mogą jednocześnie wykorzystywać jeden lub więcej mechanizmów szeregowania. Mówi się wówczas o skalowalnych systemach serwerów webowych z szeregowaniem jednostopniowym, dwustopniowym lub trójstopniowym. Nie spotyka się systemów o większej ilości stopni szeregowania. Modele systemów z szeregowaniem wielostopniowym można budować poprzez złożenie modeli prostych. Przykładowe modele złożone zostały przedstawione na rys.3.6.



Rysunek 3.6: Złożone modele systemów do równoważenia obciążeń

- Systemy z szeregowaniem jednostopniowym. Systemy z szeregowaniem jednostopniowym odzwierciedlają modele: **A, E, F, G** (rys.3.5). W modelu **A** egzekutor zintegrowany jest z serwerem DNS, natomiast w modelach**E, F, G** rolę egzekutora pełni dystrybutor lub przełącznik webowy.
- Systemy z szeregowaniem dwustopniowym. Przykładowe systemy z szeregowaniem dwustopniowym obrazują modele **H, I, J** (rys.3.6). W przypadku **H** oraz **I** [**?**, **?**, **?**, **?**]

na pierwszym stopniu wykorzystywany jest model **A**, natomiast na drugim, odpowiednio model **B** lub **D**. W obu przypadkach na pierwszym stopniu egzekutor zintegrowany jest serwerem DNS, natomiast na drugim rolę egzekutora pełni ten z serwerów webowych, który został wytypowany na poziomie serwera DNS. Przypadek **J** ukazuje kaskadowe złożenie modeli **G** (rys.3.6) w celu szeregowania żądań w systemie rozproszonych klastrów webowych [**?**]. Na obu stopniach rolę egzekutora może pełnić dystrybutor lub przełącznik webowy. Model ten charakteryzuje się krótką drogą zapytań, jednak awaria pierwszego stopnia szeregowania unieruchamia cały system rozproszonych klastrów.

- Systemy z szeregowaniem trójstopniowym. Przykład systemu z szeregowaniem trójstopniowym pokazuje model **K** [**?**]. Jest to system rozproszonych klastrów webowych, gdzie każdy z serwerów w klastrze ma wbudowany mechanizm szeregowania. Na pierwszym stopniu szeregowania wykorzystywany jest model **A**, na drugim model **G**, a na trzecim model **B**, czyli na pierwszym stopniu egzekutor zintegrowany jest z serwerem DNS, na drugim role egzekutora pełni dystrybutor lub przełącznik webowy, natomiast na trzecim ten serwer webowy, który został wytypowany przez egzekutor na drugim stopniu. Każdorazowe zadziałanie trzeciego stopnia szeregowania, może powodować znaczne opóźnienia, jednak model ten jest bardzo odporny na przeciążenia i awarie.

3.5.5 Podział ze względu na poziom szczegółowości szeregowania

Szeregowanie na poziomie serii żądań o strony

Aby zrealizować żądanie klienta o stronę umiejscowioną pod pewną nazwą domenową, mechanizm szeregowania, który jest zintegrowany z serwerem DNS, musi przydzielić tej nazwie adres IP najlepszego serwera webowego (hostname resolution). Ze względu na dużą bezwładność DNS kolejne żądania klienta o strony umiejscowione pod tą nazwą domenową będą przez pewien czas kierowane do tego samego serwera webowego. Przypadek, w którym szeregowanie żądań klientów odbywa się na poziomie serii żądań o strony, przedstawia model **A**.

Szeregowanie na poziomie żądania o stronę

W skład strony webowej wchodzi zwykle wiele obiektów. Na tym poziomie przekierowanie żądania o stronę pociąga za sobą przekierowanie serii żądań o jej elementy składowe. Mechanizm szeregowania wykorzystuje w tym celu właściwości protokołu HTTP (HTTP redirection [**?**]). Przypadki, w których występuje tego typu szeregowanie, ukazuje model **B** oraz **E**. W przypadku modelu **B**, gdy jeden z serwerów webowych otrzymuje od klienta żądanie o stronę, jeśli nie decyduje się sam zrealizować zlecenia, odsyła klientowi adres IP

lub nazwę domenową lepszego od siebie serwera [**?**]. W przypadku modelu **E**, gdy przełącznik otrzymuje od klienta żądanie o stronę, odsyła mu adres IP lub nazwę domenową najlepszego serwera [**?**]. Należy zaznaczyć, że może wystąpić tu zjawisko buforowania przez klienta adresu IP serwera, do którego nastąpiło przekierowanie. Można wówczas mówić o szeregowaniu na poziomie serii żądań o strony.

Szeregowanie na poziomie żądania o obiekt

Przypadki, w których szeregowanie odbywa się na poziomie żądania o obiekt, przedstawiają modele **C, D, F** oraz **G**. W tej grupie mechanizm szeregowania zajmuje się przekierowywaniem pakietów (packet redirection). Pakiety zawierające żądanie o obiekt muszą w komplecie trafić do wybranego przez egzekutor serwera. Jednym ze sposobów realizujących ten cel jest zapisywanie wyników decyzji egzekutora w tzw. tablicy powiązań (binding table). Ponieważ w modelu **D** ruch pakietów powracających (znacznie większy niż w przypadku pakietów z żądaniami) jest kierowany do klienta z pominięciem serwera-egzekutora, model ten jest z powodzeniem implementowany. Charakterystyczną cechą modeli **F, G** jest to, że egzekutor maskuje serwery webowe za pomocą jednego, wirtualnego adresu IP-SVA (single virtual IP address). Aby przekazać żądanie do konkretnego serwera, stosuje różne, niekiedy wyrafinowane techniki. Gdy egzekutor przekierowuje pakiety z żądaniami bez świadomości ich treści (content information blind), nazywany jest dystrybutorem (dispatcher, level 4 web-switch). Gdy egzekutor podczas podejmowania decyzji o przekierowaniu wykorzystuje informację zawartą w żądaniu (content information aware), nazywany jest przełącznikiem webowym (content switch, level 7 web-switch). Należy podkreślić, że niniejszy podział dotyczy wariantu, gdzie w modelach **F, G** rolę egzekutora pełni dystrybutor. W przypadku gdy egzekutorem jest przełącznik webowy, szeregowanie może odbywać się na poziomie żądania o obiekt, stronę oraz (dzięki umiejętności rozpoznawania znaczników cookie) serii żądań o strony.

3.5.6 Podział ze względu na poziom kontroli zapytań

Jest to podział wyodrębniający modele, w których egzekutory mają pełna kontrolę nad zapytaniami kierowanymi do systemu serwerów webowych.

- Pełna kontrola zapytań. Teoretycznie do tej grupy należą modele **B, C, D, E, F** i **G**. Klient, po otrzymaniu od serwera DNS adresu IP egzekutora, wszystkie zapytania kieruje tylko i wyłącznie do niego. W ten sposób egzekutor ma stuprocentową kontrolę nad zapytaniami. W rzeczywistości sytuacja taka zachodzi w przypadku modeli **F** oraz **G**. Warto zwrócić uwagę, że awaria lub zapchanie się egzekutora powoduje unieruchomienie całego systemu serwerów webowych.

- Częściowa kontrola zapytań. Typowym przypadkiem, w którym egzekutor sprawuje częściową kontrolę nad zapytaniami, jest model **A**. Częściowa kontrola wynika

ze specyfiki funkcjonowania systemu DNS. Również modele **B, E** należy zaliczyć do tej grupy ze względu na możliwość występowania zjawiska buforowania przez klienta adresu IP serwera, do którego nastąpiło przekierowanie. Jeśli modele **C** i **D** byłyby implementowane samodzielnie, egzekutor, którego adres IP rozgłaszałby serwer DNS, sprawowałby pełną kontrole nad zapytaniami. Jednak w przypadku modelu **D** typowym rozwiązaniem jest jego złożenie z modelem **A** (model **I** rys.3.6). W takim przypadku każdy z egzekutorów implementowanych w serwerach webowych sprawuje częściową kontrolę nad zapytaniami kierowanymi do systemu serwerów webowych.

3.5.7 Podział ze względu na poziom zaangażowania egzekutora

Egzekutor może być w różnym stopniu zaangażowany w przekierowywanie zapytań klienta. Może być również zaangażowany w proces przekazywania odpowiedzi. Rysują się tu cztery grupy modeli: bierne, zwrotne, jednokierunkowe oraz dwukierunkowe:

- Model bierny. Jest to model **A**, w którym do egzekutora w ogóle nie trafiają żądania klienta o strony czy obiekty. Egzekutor odpowiada tylko na żądania o przełożenie nazwy domenowej na adres IP najlepszego serwera (klastra) webowego, nie biorąc bezpośredniego udziału w szeregowaniu zapytań klienta.

- Modele zwrotne. Należą do nich modele **B, E** (rys.3.5). Egzekutor, po otrzymaniu żądania, zwraca klientowi adres IP lub nazwę domenową najlepszego serwera, aby ten mógł ponowić żądanie. W modelu **B** może dodatkowo zapaść decyzja o obsłużeniu żądania przez bieżący serwer. Zaangażowanie egzekutora w proces przekierowywania jest w tym przypadku minimalne.

- Modele jednokierunkowe. Należą do nich modele **D** oraz **G**[**?**, **?**, **?**]. Żądania, które osiągają egzekutor, przekierowywane są bezpośrednio do serwerów webowych. W modelu **D** może dodatkowo zapaść decyzja o obsłużeniu żądania przez bieżący serwer. Dzięki odpowiednim zabiegom technicznym serwery kierują odpowiedzi bezpośrednio do klienta. Zaangażowanie egzekutora w proces przekierowywania jest w tym przypadku średnie.

- Modele dwukierunkowe. Należą do nich modele **C** i **F** [**?**, **?**, **?**] . Wszystkie żądania, które osiągają egzekutor, przekierowywane są bezpośrednio do serwerów webowych. W modelu **D** może dodatkowo zapaść decyzja o obsłużeniu żądania przez bieżący serwer. Serwery webowe wszystkie odpowiedzi przesyłają do egzekutora, który musi je przekierowywać do klienta.

3.6 Witryny dynamiczne

Już nawet kilkudziesięcioma dokumentami World Wide Web, czyli bardzo niewielkim serwisem, trudno jest zarządzać, zmieniać i dodawać nowe fragmenty, czuwać nad poprawnością dostępnych w nim danych. Z drugiej jednak strony oprogramowanie stosowane do korzystania z World Wide Web jest bardzo proste, szeroko znane i szybko uaktualniane.

Większość informacji zapisanych na komputerach i wykorzystywanych w biznesie czy przez środki przekazu zgromadzona jest w bazach danych, w tym najczęściej w relacyjnych bazach danych. Bazy danych pozwalają na kontrolę nad danymi, analizowanie ich, ogranizację i prezentację użytkownikowi w jak najbardziej prosty i skuteczny sposób. Potrzebny jest jednak do nich dostep przez wyspecjalizowane oprogramowanie. W systemach klient/serwer jest to klient bazy danych; kolejny program, który musi poznawać użytkownik, program nie zawsze prezentujący najwyższy poziom możliwości technicznych czy łatwości komunikacji z użytkownikiem.

Oba te systemy, World Wide Web i bazy danych, stworzone są w tym samym celu – efektywnego dostępu do informacji. Już po kilku latach rozwoju World Wide Web zorientowano się, że można je połączyć, biorąc z każdego, co najlepsze. Połączyć łatwość dostępu, jaką niesie przeglądarka World Wide Web z wysokim stopniem organizacji oferowanym przez relacyjną bazę danych.

Sam serwer HTTP nie potrafi się jednak bezpośrednio porozumiewać z bazą; potrzebuje dodatkowego oprogramowania. Dawniej najlepszą metodą nawiązania kontaktu między serwerem HTTP a bazą było napisanie programu w standardzie CGI [5]. Program CGI przyjmuje informacje od serwera HTTP, przetwarza je i wysyła do bazy danych, baza danych wykonuje zlecane jej zadanie, wysyła efekt do programu CGI, który z kolei zwraca go przez serwer HTTP do klienta HTTP. Rezultatem jest najczęściej plik HTML. Dziś mechanizm porozumiewania się pozostaje taki sam, ale w miejsce programu CGI wchodzi program napisany w API serwera HTTP, na przykład ISAPI dla Microsoft Internet Information Server. To drugie rozwiązanie jest bardziej efektywne i pozwala na mniejsze obciążenie komputera, na którym działa serwer, niż w przypadku użycia programu CGI.

Można stworzyć serwis World Wide Web, który porozumiewa się z bazą danych. Można wpisywać do bazy nowe informacje, na przykład wypełniając ankietę marketingową, można przeglądać istniejące informacje, poprawiać je o ile mamy do tego prawo a także wyszukiwać potrzebne informacje.

Dokładnie takie same możliwości ma serwis wewnętrzny firmy – intranetowy. Może on służyć do porozumiewania się z firmą oddziałów terenowych czy pracowników w podróżach służbowych, a także do zwykłej, codziennej komunikacji z bazą danych. Przeglądarka World Wide Web jest łatwiejszym w obsłudze i bardziej znanym użytkownikom oprogramowaniem niż specjalny klient bazy danych.

Do stworzenia takiego systemu potrzebny jest serwer HTTP i baza danych, a pomiędzy

[5] ang. *Common Gateway Interface*

nimi program pośredniczący. Można napisać własne oprogramowanie pośredniczące, ale również skorzystać z gotowych produktów.

3.7 Testowanie serwerów WWW

Jak wiadomo słaba wydajność wprost przekłada się na niezadowolenia klientów, a niezadowolony klient opuści witrynę Web i może nigdy już na nią nie wrócić [**?**, **?**, **?**].

Przewidywanie jak witryna WWW będzie odpowiadać na specyficzne obciążenie jest wielkim wyzwaniem. Od czasu jak sajty webowe stanowią kompleksowe systemy zawierające elementy sprzętowe, programowe i sieciowe pochodzącymi od różnych producentów, oraz posiadają bardzo różne profile wydajnościowe – jest praktycznie niemożliwa predykcja, jak dany system zachowa się podczas obciążenia. Jedynym pewnym sposobem sprawdzenia skalowalności systemu jest przeprowadzenie testów wydajnościowych, w których natężenie i charakterystyka przewidywanego ruchu jest symulowana tak realistycznie jak to jest możliwe.

Testowanie obciążenia witryn webowych stanowi priorytet dla firm robiących interesy online. Jak wielu użytkowników z akceptowalnymi czasami odpowiedzi może obsłużyć strona ? Jest to niezwykle ważna informacja wykorzystywana do planowania kampanii reklamowych, estymacji budżetów branży IT a nawet do zwykłego dostarczania usług. I pomimo wagi tego problemu praktycznie większość testów obciążeniowych jest wykonywana niepoprawnie, ponieważ nie odpowiadają one rzeczywistym warunkom.

Poniżej zostaną opisane elementy, niezbędne podczas konstrukcji wysoce realistycznych i dokładnych testów wydajnościowych sajtów webowych (znajdują się również najczęściej popełniane w typowych testach obciążeniowych błędy oraz sposób radzenia sobie z nimi):

1. Zrozumienie natury obciążenia

   Pierwszym krokiem jest dokładne i obiektywne zrozumienie natury obciążenia, jakie musi zostać wygenerowane, podczas symulacji ruchu na witrynie.

   Niestety, testowanie obciążenia sajtów webowych jest dziedziną ralatywnie nową (toteż słabo zrozumianą i udokumentowaną), a do tego jest bardzo ezoteryczną dziedziną testów. Żeby odpowiednio zrozumieć naturę obciążeń należy możliwie często korzystać z narzędzia dokonującego analizy logów naszego web serwera. Daje to możliwość obserwacji detali odwiedzin każdego użytkownika: jego adresu IP, daty i czasu odwiedzin; jakie, ile i o jakiej wielkości dane pobierał, czy pobranie było zakończone sukcesem, czy nie oraz inne dane o sesji użytkownika, jego systemie operacyjnym i narzędziu z jakiego korzystał.

   Sesja użytkownika – niezbędne informacje

Większość potrzebnych informacji jakie należy wyekstrahować podczas analizy logów są informacje związane z sesjami użytkowników[6]. Testy obciążeniowe najbardziej związane są właśnie z sesjami użytkownika. Sesję użytkownika definiuje się jako sekwencję pobrań stron związaną z unikatowym (pojedynczym, identyfikowalnym) użytkownikiem.

Typowa witryna webowa jest wizytowana przez szeroką gamę użytkowników z szeroką gamą działań. Np. użytkownicy stron związanych z szeroko rozumianym e–commersem: niektórzy z nich przyszli pooglądać (*browse*), niektórzy kupić, jeszcze inni sprawdzić status zamówień. Nawet jeśli grupa klientów dokonuje pojedynczego działania, takiego jak kupowanie książki, każdy z tej grupy może to realizować na szereg sposobów. Niektórzy będą poruszać się ze strony na stronę bardzo szybko, raczej nie dbając o dokładne przeczytanie znajdujących się tam informacji, inni przeciwnie – poruszają się powoli czytając każdą stronę bardzo uważnie. Niektórzy będą wyszukiwać czytając fragmenty wielu książek zanim zdecydują się na kupno jednej, inni skierują się od razu do miejsca zamówienia *(purchase page)*.

Zrozumienie tej szerokiej gamy akcji, działań i zachowań jest niezbędne (krytyczne) dla zaprojektowania testów obciążeniowych, ponieważ dobrze zaprojektowany test powinien odzwierciedlać zachowania użytkowników tak precyzyjnie jak to tylko możliwe. Najlepszą metodą jest używanie analizatorów logów (podczas pracy witryny) oraz wydobycie kluczowej grupy zmiennych zachowania klientów najczęściej spotykanych na danym sajcie, adekwatnych do ruchu webowego. Niektóre ze zmiennych, na które zawsze powinno zwrócić się uwagę to:

- długość trwania sesji (mierzona w stronach);
- czas trwania sesji (duration) (mierzona w minutach i sekundach);
- typ stron wizytowanych podczas trwania sesji (strona domowa, strona informacji o produkcjie, strona informacji o kartach kredytowych).

Naturalnie nie są to wszystkie zmienne mające wpływ na charakterystykę obciążeniową sajtu – oprócz nich należy wybrać jeszcze jakieś zmienne które o specyfice danej witryny decydują.

Taką specyfikę można zauważyć na przykładzie: siedmio–stronicowa sesja, której rezultatem jest zamówienie artykułu znacznie bardziej obciąża system niż siedmiostronicowa sesja, której wynikiem jest tylko przeglądanie dokumentów. Przeglądanie dokumentów jest zwykle związane z dokumentami statycznymi, zaś na sesję, której efektem jest dokonanie zakupu artykułu składa się cały szereg czynników takich jak: przeszukiwanie bazy danych zasobów

[6]ang. *user sessions*

witryny, użytkownika, transakcji kart kredytowych z weryfikacją (osobne systemy), a także wysłanie potwierdzającego zamówienie e–maila. Statystycznie rzecz biorąc pojedyncza sesja zakupu może pochłonąć zasoby sajtu tak jak dwadzieścia sesji statycznych *(browsing)*.

W podobny sposób można porównać zakupy dokonywane przez nowych i stałych klientów. Nowy użytkownik potrzebuje dokonać rejestracji, potwierdzenia i weryfikacji swoich danych, zaś stały już nie. Związane z bazą danych zakładanie użytkownika może być równe obciążeniu generowanemu przez pięciu stałych użytkowników, dlatego trzeba wyróżnić co najmniej dwa typy wykorzystania zasobów przy realizacji zakupów.

Gdy zostaną już zdeterminowane wszystkie zmienne związane z sesją użytkownika – należy następnie znaleźć (zwykle również w logach) rangę i dystrybucję wartości tych zmiennych np. procentowa (w całości żądań) ilość żądanych stron podczas sesji. Do takich celów można używać narzędzi statystycznych takich jak: odchylenie standardowe, jednakże najlepszym sposobem charakteryzowania większości zmiennych webowych testów obciążeniowych jest użycie rozproszenia jako wartości dyskretnej.

Wielkość detali i precyzję z jaką należy analizować te zmienne zależy od struktury sajtu (i jej komplikacji), czasu testów, oraz analizy wyników.

Współbieżni użytkownicy

Zazwyczaj w testach obciążeniowych witryn webowych podaje się wartości obciążenia spowodowanego naraz współużytkującymi zasoby użytkownikami. Jednakże współbieżni użytkownicy nie powinni być widziani jako zmienna wejściowa w teście, ale jako rezultat wielu różnych czynników. A zwkle podaje się np. strona została przetestowana z obciążeniem 1000 współbieżnymi użytkownikami. Liczba współbieżnych użytkowników nie jest miernikiem obciążenia. Wartość ta jest rezultatem wydajności (zdolności do przyjęcia obciążenia) witryny. Efektem pracy na wolniejszym sajcie będzie większa liczba współbieżnych użytkowników. Można to pokazać na przykładzie – jeśli na stronę loguje się trzech użytkowników w odstępnie czasowym 1min i pracują po 1 min (np. dokonują zakupów) – to tylko i wyłącznie w zależności od wydajności witryny oraz jej obciążenia będzie 0, 2 i 3 naraz pracujących użytkowników. Zatem jeśli wydajność witryny webowej z jakiegoś powodu spada to wzrastać będzie liczba współbieżnych użytkowników. Trzeba jednakże dodać jeszcze jeden element – jeśli wydajność sajtu spada (rośnie liczba współbieżnych użytkowników) rośnie także ilość użytkowników, którzy rezygnują z korzystania z jego zasobów. Jeśli witryna jest bardzo szybka to mogą się zdarzyć sytuacje, że ilość współbieżnych użytkowników będzie oscylować wokół zera (nawet przy ich dużej ilości).

Z tych powodów używanie jako miernika obciążenia strony ilości współpra-

cujących naraz użytkowników jest (jeśli ma być realistyczne) bardzo trudne, a wyniki zawsze odbiegają od rzeczywistych. Znacznie lepszym wskaźnikiem obciążenia witryny jest liczba sesji użytkownika wystartowana na godzinę. Ma to ważną zaletę – jest to wartość stała, niezmienna w zależności od wydajności witryny w teście. Daje wskaźnik ilu użytkowników otrzyma stronę pierwszą witryny – to czy ci użytkownicy ukończą swoją sesję, czy nie zależy już od zdolności strony do uniesienia danego obciążenia. I jest to właśnie ta wartość, którą się poszukuje podczas wykonywania testów.

Rezygnacja klientów

Kolejną ważną wielkością, niejednokrotnie źle szacowaną w testach, a mającą znaczny wpływ na testy obciążeniowe witryny jest rezygnacja klientów w trakcie sesji. Użytkownicy często rezygnują podczas sesji np. zakupów z powodu bardzo długich odpowiedzi serwera. Zmienną tą należy również brać pod uwagę podczas wykonywania testów obciążeniowych na przygotowywanym sajcie. Symulacja porzucania przez użytkowników sesji powinna być dokonana tak rzeczywiście jak to tylko możliwe. Jeśli nie, podczas testów będzie się symulować obciążenie tak wielkie jakie może nigdy nie nastąpić, lub wąskie gardła, które w rzeczywistości nigdy się nie pojawią. Pominie się zatem najbardziej istotne wyniki testów obciążeniowych: czyli użytkowników, którzy mogą porzucić sesję z powodu słabej wydajności. Zatem testy staną się nieużyteczne. W poniższej tabelce znajdują się przykładowe zależności pomiędzy liczbą opuszczających witrynę użytkowników, a czasem oczekiwania na określoną stronę. Jeśli zatem odpowiedź oczekiwania na stronę domową

Typ strony	% opuszczających 0-5 s	% opuszczających 5-10 s	% opuszczających 10-15 s	% opuszczających 10-20 s
Strona domowa	0%	30%	45%	75%
Stock quote	0%	15%	25%	45%
Stock Transaction	0%	0%	0%	15%
Account Information	0%	5%	15%	35%

Tablica 3.1: Średni stopień porzucania witryny w % dla różnych typów stron

wynosi 5 sek. lub mniej nie obserwuje się ucieczki klientów. Jednakże powyżej tego czasu rezygnacja z oczekiwania staje się coraz wyraźniejsza, by w czasie 15–20 sek. $3/4$ obecnych na witrynie klientów już zrezygnowało. Jednakże należy zwrócić uwagę na jeszcze jeden związany z tym szczegół. Rezygnacja użytkowników zmniejsza obciążenie, zatem wydajność zaczyna się zwiększać, mniej użytkowników rezygnuje – aż w końcu obciążenie z powrotem powoduje oczekiwania na realizację żądań o strony i cykl się powtarza. Jest to oczywiście przykład. Rezygnacja użytkowników z usług witryny jest bardzo ważną zmienną pokazującą, że dany sajt nie jest w stanie satysfakcjonująco zapewniać usług przy obciążeniu. Najczęściej oprogramowanie do testowania potrafi

symulować rezygnację użytkowników, ale z ekstremalnych powodów – zwykle 60–120sek. Jednakże takie wartości są nie do przyjęcia gdyż praktycznie nikt nie czeka tyle czasu. Aby móc wykonać testy obciążeniowe tak prawdopodobne jak to tylko możliwe, można skonfigurować testowaną witrynę w taki sposób, aby przekierowywać część użytkowników na wolniejszy mirror tej witryny. Można np. w ten sposób: 90% ruchu jest kierowane na zwykły serwer, zaś 10% na wolniejszy mirror, na którym np. strona domowa będzie serwowana później o 5 sek. W takiej konfiguracji należy ten dwuserwerowy system uruchomić na kilka godzin lub dni, do czasu otrzymania znaczących wyników. Po tym czasie należy dokonać analizy porzucających transakcje webowe klientów. Jeśli na zwykłym serwerze opuszczanie sesji po stronie domowej sięga 6%, zaś na tym wolniejszym 20%, trzeba się liczyć z opuszczaniem witryny przez 14% klientów, którzy nie będą zbyt cierpliwi by poczekać kolejne 5 sek.

Porzucanie operacji webowych przez klientów nie jest interesującym wynikiem, samym w sobie – daje pogląd na to jakie części sajtu są najbardziej obciążone w warunkach pracy. Jeśli np. strona domowa jest wyjątkowo wolna, większość użytkowników nawet nie rozpocznie sesji. Widać zatem, że odpowiednie symulowanie tego parametru jest bardzo istotne dla stworzenia wydajnego sajtu.

Dystrybucja zapytań o stronę

Kolejną ważną zmienną, której warto poświęć uwagę jest dystrybucja zapytań o stronę. Ta ważna wartość określa o jakie strony są zapytania i w jakich proporcjach. Procedura zbierania tej danej jest dwustopniowa: najpierw należy zdefiniować skategoryzowane grupy stron, a następnie obliczyć udział procentowy żądań o strony w każdej grupie.

2. Estymacja wzrostu ruchu na witrynie

   Kolejnym krokiem przy projektowaniu testów obciążeniowych jest zrozumienie pewnych kluczowych zmiennych – niezbędnych do estymowania docelowego poziomu obciążenia:

   - jak wzrasta całkowite obciążenie ruchu na sajcie;
   - jaki jest poziom obciążenia pojedynczych pików, mających wpływ w całkowitym obciążeniu;
   - jak szybko liczba użytkowników może wpłynąć na osiągnięcie maksimum piku obciążeniowego;
   - jak długo trwają poszczególne piki.

   Przy pomiarze poziomu obciążenia można używać zmiennej: liczba sesji użytkowników na jednostkę czasu. Taki wybór wynika z jego prostoty w zrozumieniu i łatwości

analizy.

Estymacja przyszłego ruchu webowego

Ocena szybkości wzrostu całkowitego obciążenia ruchu jest istotna, ponieważ rozmiar poziomu pików jest proporcjonalny do amplitudy całkowitego ruchu. Jeśli całkowity ruch wynosi np. 100000 sesji na tydzień, oznacza to piki o wielkości 1500 sesji na godzinę, zatem przy obciążeniu rzędu 200000 sesji na tydzień piki (chwilowe obciążenia) mogą sięgać 3000 sesji na godzinę.

Całkowity wzrost wielkości ruchu wynika z dwóch czynników: danych historycznych (o wzroście) oraz szacunków sprzedaży/marketingu. Aby estymować te wartości, należy dokonać analizy tygodniowego ruchu oraz dowiedzieć się od odpowiednich ludzi w firmie jak może się zmieniać ruch z miesiąca na miesiąc, oraz jak może wyglądać zainteresowanie na witrynie po specjalnych ofertach marketingowych. Z takich informacji można oszacować wartość obciążenia w czasie – czyli przyszłe możliwości sajtu.

Estymacja ruchu chwilowego

Po dokonaniu analizy wzrostu całkowitego obciążenia, należy estymować poziom natężenia ruchu chwilowego. Jest to niezbędne ponieważ ruch webowy jest raczej nierównomierny, a wiele sajtów doświadcza znaczących chwilowych obciążeń. Zwykle zdarza się to tylko kilka razy (raz, lub dwa w tygodniu lub kilka godzin dziennie). Gdy ruch webowy jest najwyższy. Przykładem są witryny pogodowe, na których nasilenie ruchu odbywa się w piątek i sobotę, ponieważ użytkownicy potrzebują danych pogodowych z powodu planów weekendowych. Natomiast (trading) handel sieciowy zwykle doświadcza największych obciążeń w okolicach czasu otwarcia i zamknięcia sklepu.

Szacowanie przebiegu ruchu chwilowego

Estymowanie jak szybko osiągane jest docelowe chwilowe obciążenie, oraz przez jak długi czas jest utrzymywane jest tak ważne jak estymacja amplitudy chwilowych obciążeń. Można to pokazać na przykładach:

Sprzedaż online (stock) doświadcza zwykle ekstremalnie ostrych chwilowych natężeń ruchu (pików) w różnym czasie jak np. otwarcie sklepu. W ciągu kilku minut witryna przechodzi od nie przyjmowania zgłoszeń po przyjmowanie ich tysięcy naraz. Test obciążeniowy powinien dla tego typu sajtów generować tak specyficzne wysokie chwilowe obciążenia zwykle w czasie od pięciu do dziesięciu minut.

Sprzedaż ubrań online, może doświadczać innej charakterystyki obciążeniowej. Tutaj testy obciążeniowe powinny generować wzrost obciążenia (do docelowego maksymalnego) w czasie rzędu jedna dwie godziny. Szybszy wzrost może odbiegać od sytuacji rzeczywistej.

Czas trwania pików obciążeniowych jest również bardzo ważny – witryna radząca sobie z wysokim obciążeniem trwającym pięć do dziesięciu minut może się zupełnie załamać podczas dłużej trwającego wysokiego ruchu.

3. Dokumentowanie i projekt

   Gdy wszystkie powyżej wymienione informacje zostaną zebrane można zaprojektować odpowiedni do danych warunków test obciążeniowy.

   Kluczowymi elementami takiego testu są:

   - cele testu: zdeterminowanie jakiemu obciążeniu oraz w jaki sposób będzie poddawana witryna, a także czy wraz z wzrastającym ruchem jest w stanie mu podołać;
   - kryteria jakie będą spełnione podczas trwania klienckiego żądania – tzn. np. maksymalny czas oczekiwania na stronę lub nawet czas po jakim żądanie może zostać odrzucone;
   - opis skryptów testowych i ich typy: np. scrypt opisujący trójstronicowe żądanie (strona domowa » informacja o produkcie » zamowienie) oraz procentowe wykorzystanie tych typów skryptów;
   - opis scenariusza czyli użycie skryptów testowych, czas ich uruchamiania oraz ich udział procentowy w całości testu, a także czas i szybkość narastania obciążenia (piki obciążeniowe).

Rozdział 4

Zarządzanie wielokomputerowym serwisem WWW

W rozdziale tym znajdzie się próba odpowiedzi na pytanie co to jest kiedy i po co stosuje się zarządzanie serwerem WWW (jakie są parametry charakterystyczne oceny wydajności i dostępności). Następnie zostanie przedstawiona szczegółowa taksonomia serwerów WWW. Ostatnim punktem tego rozdziału będzie porównanie usług zapewnianych przez witryny statyczne i dynamiczne, opis zastosowań i możliwości obu typów tworzenia witryn oraz wymagania stawiane systemom WWW wraz z poglądowym przykładem.

4.1 Wprowadzenie

Jak zauważono wcześniej – przyszłość serwisów WWW należy do platform wieloserwerowych. Wynika to z możliwości teoretycznie niograniczonej rozbudowy wraz ze wzrostem liczby użytkowników oraz ich wymagań. Kolejną zaletą jest również spełnianie bezpieczeństwa przez takie wielokomputerowe systemy – w dowolnym elemencie takiego systemu – nodzie – może nastąpić praktycznie dowolnego rodzaju uszkodzenie – tak dysk, jak pamięć operacyjna czy procesor – wtedy poszczególne procesy są transportowane na inne nody, nawet w przypadku uszkodzenia jednej maszyny, nie spowoduje to przestoju. Jednakże aby taki system spełniał dobrze swoje zadania potrzebne jest oprogramowanie lub urządzenie pozwające nim zarządzać czyli odpowiednio rozdysponować żądania klienckie oraz wykrywać uszkodzenia poszczególnych elementów układu. W literaturze przyjęło się takie rozdysponowanie zapytań pomiędzy poszczególne nody nazywać równoważeniem obciążenia systemu[1].

[1] ang. *load balancing, LB*

4.2 Równoważenie obciążenia – metody

Równoważenie obciążeń (ang. *load balancing*) w systemach rozproszonych, to zagadnienie z dziedziny rozdziału zadań polegające na dystrybucji pomiędzy węzły systemu napływających do niego zleceń tak, aby maksymalizować jego łączną wydajność. Oznacza to, że działalność związana z rozdziałem zadań nie może powodować obciążenia systemu w stopniu, który niwelowałby korzyści wynikające z faktu zrównoważenia obciążeń jego węzłów. Angielski termin load balancing w rzeczywistości funkcjonuje jako reprezentant tej problematyki, gdyż traktowany ściśle oznacza doprowadzenie systemu rozproszonego do stanu w którym wszystkie jego węzły obciążone są w dokładnie równym stopniu, nawet jeżeli oznaczałoby to odebranie zadań jednemu z nich. W stosunku do większości rozwiązań przemysłowych z tego zakresu należałoby poprawnie używać określenia współdzielenie obciążeń (ang. *load sharing*) lub bardziej ogólnie, rozdział obciążeń (ang load distribution). Realizacja ścisłego równoważenia obciążeń wiąże się z koniecznością implementacji metod odbierania zadań węzłom, powoduje to wzrost złożoności realizacji algorytmu, a sam proces odbierania zadania jest dość kosztowny. Wzrost wydajności uzyskiwany dzięki idealnemu zrównoważeniu obciążeń w systemie w porównaniu do wydajności uzyskiwanej przy zastosowaniu współdzielenia obciążeń nie jest zwykle na tyle duży, by usprawiedliwiał ponoszenie kosztów związanych z pokonaniem złożoności realizacji ścisłego równoważenia obciążeń [**?**, **?**, **?**]. W tej pracy, tak jak w literaturze angielskojęzycznej, używany jest termin równoważenie obciążeń, chyba że zaznaczono inaczej. Wydaje się, że najskuteczniejszą metodą zwiększenia wydajności serwera WWW jest powielenie (lub podział) danych, które serwer udostępnia, pomiędzy grupę połączonych siecią komputerów (dodatkowych serwerów WWW) i zapewnienie dostępu do tej grupy tak jak do pojedynczego serwera z uwzględnieniem transparentnego dla użytkownika i maksymalizującego wydajność podziału żądań HTTP pomiędzy poszczególne elementy grupy. Taka grupa nazywana klastrem lub farmą serwerów stanowi swoisty system rozproszony, którego wydajność można maksymalizować stosując równoważenie obciążeń. Należy zaznaczyć, że chociaż termin klaster oznacza serwery połączone w całość logiczną (dostępne pod jedną nazwą lub adresem IP), nie implikując żadnych ograniczeń na geograficzne rozproszenie serwerów, to wiele rozwiązań komercyjnych wymaga, by serwery należące do klastra znajdowały się w obszarze jednej sieci lokalnej (miały jednakową część globalną adresu IP).

4.3 Przegląd i podział webowych algorytmów szeregowania

Wszystkie mechanizmy realizujące zarządzanie dostępem do rozproszonych serwerów WWW muszą mieć zaimplementowany konkretny algorytm, na podstawie którego będą mogły po-

dejmować decyzję o przesłaniu zapytania do najlepszego serwera. W zależności od stopnia skomplikowania systemu i możliwości implementacyjnych wykorzystywane algorytmy mogą mieć różną złożoność obliczeniową. Zastosowane strategie zależne są również od informacji, jakie posiada system na temat ilości zapytań generowanych przez klientów oraz informacji o stanie serwerów wchodzących w skład rozproszonego systemu WWW. Można je podzielić na kilka kategorii w zależności od ilości informacji, jaka wykorzystywana jest przy podejmowaniu decyzji.

4.3.1 Algorytmy statyczne (nie wykorzystujące informacji zewnętrznych)

Jeżeli w procesie szeregowania nie są wykorzystywane żadne informacje o stanie serwerów lub strumieniu zapytań, mówi się o algorytmach statycznych. Te mechanizmy, to najprostsze rozwiązania gwarantujące w niewielkim stopniu rozłożenie obciążenia pomiędzy serwery. Rozkład ten jest bardzo przypadkowy i nie gwarantuje równomiernego wykorzystania zasobów ani wysokiej dostępności systemu, jednakże są one najszybszymi, gdyż wymagają najmniej mocy obliczeniowej. Do takich algorytmów należą:

- Random – system nie posiada żadnych informacji o stanie serwerów WWW, jak również nie posiada informacji o tym, do którego serwera ostatnio skierowane zostało zapytanie;
- Round–robin – system nie posiada żadnych informacji o stanie serwerów WWW, posiada jednak informację historyczną o tym, do którego serwera skierowane zostało ostatnio nadesłane zapytanie.

4.3.2 Algorytmy dynamiczne (wykorzystujące informację zewnętrzną)

W procesie szeregowania można stosować bardziej rozbudowane mechanizmy zarządzania. Aby zwiększyć efektywność szeregowania, można stosować strategie wykorzystujące informacje o kliencie. Wybór najlepszego serwera może odbywać się na podstawie adresu IP lub numer portu TCP wykorzystywanego przez klienta. Możliwe jest również wykorzystywanie tzw. alarmów generowanych przez serwery. Do mechanizmu decydującego o wyborze serwera w określonych odstępach czasowych przekazywana jest informacja zwrotna o obciążeniu serwera lub informacja o przekroczeniu jakiegoś określonego parametru. Szeregowanie może odbywać się na podstawie jednej z metryk określających obciążenie:

- ilości aktywnych połączeń z serwerem,
- wykorzystania dysku i/lub procesora serwera,

- przewidywanego strumienia zapytań w określonym czasie.

W przypadku modeli, gdzie mechanizmy szeregowania w całości kontrolują strumień zapytań, mogą być stosowane algorytmy dystrybuujące zapytania w zależności od wielkości napływającego strumienia oraz obciążenia serwerów obsługujących te zapytania. Dzięki informacjom zbieranym od serwerów możliwe jest szacowanie parametrów mających wpływ na szybkość ich odpowiedzi. Mogą to być:

- ilość zapytań zrealizowana w określonym czasie,
- obciążenie procesora w danym momencie,
- poziom wykorzystania dysków w serwerze.

Takie rozwiązania umożliwiają wyeliminowanie przeciążonych serwerów. W znacznym stopniu zwiększa to dostępność klientów do danych oraz wpływa na lepsze wykorzystanie zasobów.

Algorytmy dynamiczne możemy w zależności od miejsca analizy podzielić na:

- *Server Info Aware* – czyli wykorzystujące informacje o stanie serwera;
- *Client Info Aware* – czyli wykorzystujące specyfikę żądań klienckich – przekierowujące żądania w zależności od typu zapytania klienckiego (dane z serwera baz danych, wykorzystujące skrypty CGI, zawierające pliki multimedialne itp.)

4.3.3 Algorytmy szeregowania wykorzystywane po stronie serwera DNS

Algorytmy nie wykorzystujące informacji o stanie systemu – statyczne

- Round–robin DNS
  Podejście to zostało zastosowane jako pierwsze przy budowie skalowalnego systemu serwerów webowych przez NCSA [**?**]. Kod serwera DNS (np. Unixowy BIND) bez żadnych modyfikacji może wykorzystywać algorytm Round–robin. Obciążenie serwerów nie jest zbyt dobrze równoważone ze względu na mechanizm cache dla powiązań nazwa domenowa - adres IP. Stosując takie podejście, nie ma się również kontroli nad dostępnością serwerów oraz nie uwzględnia się możliwości zastosowania serwerów o różnej wydajności. Jest to algorytm bardzo prosty zarówno w działaniu jak i implementacji (w stadardowym serwerze DNS można go zastosować, bez konieczności modyfikacji kodu serwera). Jego działanie polega na cyklicznym rozsyłaniu pakietów do kolejnego na liście serwera. Oczywiście najlepsze wyniki osiąga się dla komputerów symetrycznych sprzętowo. Modyfikacją tego algorytmu jest Weighted Round–Robin, w którym to algorytmie można statycznie nadać wagi poszczególnym serwerom w zależności od ich wydolności;
- Random

Algorytmy wykorzystujące informacje o stanie systemu

Alternatywą dla algorytmu Round–robin najczęściej stosowanego w przypadku serwera DNS są algorytmy wykorzystujące informacje o stanie systemu (wielkości przewidywanego strumienia zapytań, aktualnego obciążenia serwerów). Okazuje się, że reguły bazujące wyłącznie na stanie obciążenia serwerów WWW są nieefektywne. W rzeczywistości, w związku z hierarchicznym buforowaniem danych przez serwery DNS, pojedyncze zapytanie od klienta może spowodować napływ wielu innych zapytań. Wynika z tego, że informacja o aktualnym obciążeniu nie jest w żaden sposób związana z przyszłym obciążeniem [**?**]. Podejście pozwalające oszacować nadchodzący strumień zapytań (*hidden load weight*) w czasie TTL (*Time To Live*) jest najbardziej efektywne. Dzięki tym informacjom można przypisać różne wartości TTL dla różnych domen i estymować wielkość ukrytego, czyli niekontrolowanego przez serwer DNS strumienia zapytań. Przykładami tego typu algorytmów są [**?**]:

- Multi tie round–robin – dla każdej domeny lub ich określonych zbiorów przypisywane są różne wartości TTL, pozwalające na zrównoważenie strumieni zapytań do każdego serwera WWW.
- Dynamically accumulated load (DAL) – rozbudowana wersja algorytmu Multi tie Round–robin polegająca na zbieraniu informacji o estymowanej wielkości obciążenia serwerów i zmianie łańcucha przypisań adresów w serwerze DNS w zależności od zakładanego obciążenia.
- Minimum residual load – modyfikacja algorytmu DAL polegająca na estymowaniu wielkości ukrytego obciążenia i wartości TTL. Po upływie TTL serwer WWW odpytywany jest o rzeczywistą wielkość strumienia zapytań, jaka została do niego skierowana. Dzięki temu algorytm ma informację o stanie serwera. Powoduje to również zmianę kolejności przypisań w serwerze DNS;

Algorytmy adaptacyjne

Algorytmy równoważenia bazujące na oszacowaniu hidden load weight oraz alarmach z krytycznie przeciążonych serwerów pozwalają na dużo bardziej efektywne, w porównaniu z algorytmami Round–robin, równoważenie serwerów WWW. Niestety, są one efektywne tylko w przypadku rozproszenia serwerów homogenicznych. Inną grupą algorytmów przeznaczonych do równoważenia obciążenia serwerów heterogenicznych są algorytmy adaptacyjne. W przypadku algorytmów adaptacyjnych wartość TTL przypisywana jest dynamicznie do każdego żądania na podstawie przewidywanego hidden load weight oraz wydajności serwera, do którego zapytanie zostało przypisane. Idea tego podejścia polega na uwzględnieniu pewnej wartości ξ_i opisującej moc obliczeniową serwera S_i.

Algorytmy te podzielone zostały na dwie grupy [**?**]:

- probabilistyczne – realizacja metody bazuje na algorytmie Round–robin. Za każdym razem losowo generujemy liczbę $\gamma(0 \leq \gamma \leq 1)$. Jeżeli zapytania przypisane są aktualnie do serwera S_i, to jako następny zostanie wybrany S_{i+1} tylko wtedy, gdy $\gamma \leq \xi_i$, w przeciwnym przypadku warunek zostaje rozpatrzony dla kolejnego serwera. Wartość TTL oblicza się ze wzoru:

$$TTL_i = \frac{\eta}{\xi_i} \tag{4.1}$$

- deterministyczne – ideą algorytmu jest takie dostosowanie czasu TTL, aby bardziej wydajne serwery obsługiwały więcej zapytań, a mniej wydajne nie były przeciążane. Dla każdej domeny i obsługiwanej przez system serwerów webowych zostaje przypisana wartość TTL z uwzględnieniem wydajności serwera j według wzoru:

$$TTL_{ij} = \frac{\eta * c_i}{\xi_i} \tag{4.2}$$

  gdzie η jest wartością stałą, zależną od liczby zdefiniowanych domen generujących zapytania.

4.3.4 Algorytmy szeregowania wykorzystywane w dystrybutorach

Stosowane tutaj algorytmy [**?**] są najprostsze, ponieważ dystrybutor obsługuje wszystkie nadchodzące pakiety i konieczne jest zminimalizowanie czasu poświęcanego na ich obsługę. Rekompensatą za konieczność obsługi wszystkich pakietów jest pełna kontrola napływającego strumienia zapytań. Stosowane w tym przypadku algorytmy można podzielić na trzy podstawowe grupy:

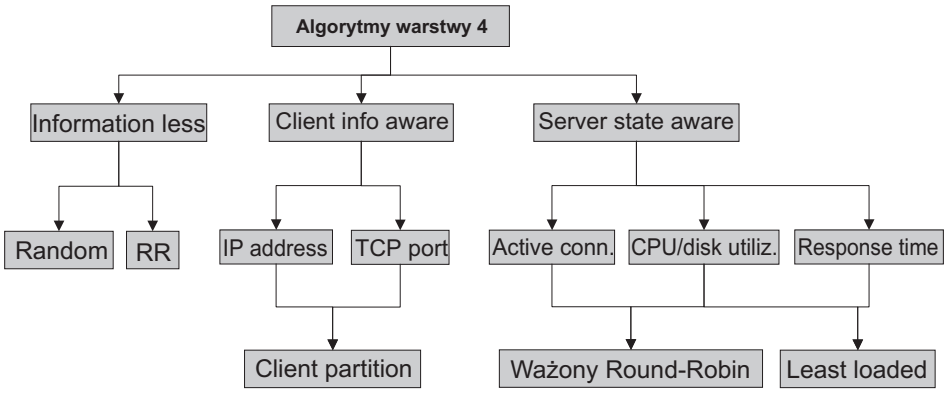
- Information less – nie wykorzystujące żadnych informacji o stanie systemu,
- Client info aware – wykorzystujące informacje o adresie IP klienta lub numerze portu, przez który komunikuje się z serwerem,
- Server state aware – wykorzystujące informacje o stanie serwera, do którego przekazują zapytanie.

Tak jak w przypadku algorytmów stosowanych w serwerze DNS, najmniej efektywne są algorytmy nie wykorzystujące żadnych dodatkowych informacji, czyli Random i Round–robin.

Wykorzystując informacje o kliencie, np. jego adres IP, można zastosować bardziej efektywne rozwiązania. Przykładem takiego algorytmu jest Client partition.

W związku z zarządzaniem strumieniem zapytań na poziomie pakietów, odwołania nadchodzące z tego samego adresu, dotyczące pojedynczej sesji, muszą być przekierowywane do tego samego serwera. Dopiero żądania nowego obiektu mogą być przesłane na inny serwer. Wymaga to utrzymywania w systemie tablicy aktywnych połączeń. Dzięki

temu można oszacować, który z serwerów obciążony będzie dodatkowymi zapytaniami w najbliższym czasie i na tej podstawie wybrać najlepszy, do którego skierowane zostanie kolejne zapytanie. Realizowane jest to na podstawie algorytmu Least loaded server.



Rysunek 4.1: Algorytmy stosowane w dystrybutorach.

Oprócz informacji o tym, kto przesyła zapytania do systemu webowego, można również uzyskać dane na temat pracy poszczególnych serwerów w systemie. Aby poprawić jakość szeregowania zapytań, w algorytmie dokonującym wyboru najlepszego serwera stosuje się różnego rodzaju metryki określające aktualne obciążenie serwerów. Najczęściej stosowanymi algorytmami w tym przypadku są Least loaded server oraz Weighted Round–robin.

- W przypadku algorytmu Least loaded server dzięki wybranemu kryterium (określonej metryce) wiemy, który z serwerów powinien przyjąć kolejne zapytanie. Odbywa się to na zasadzie: najmniej obciążony serwer przyjmuje kolejne zlecenie.
- Stosując algorytm Weighted Round–robin, przy wyborze kolejnego serwera można uwzględniać metrykę jako parametr funkcji wyboru najlepszego serwera.

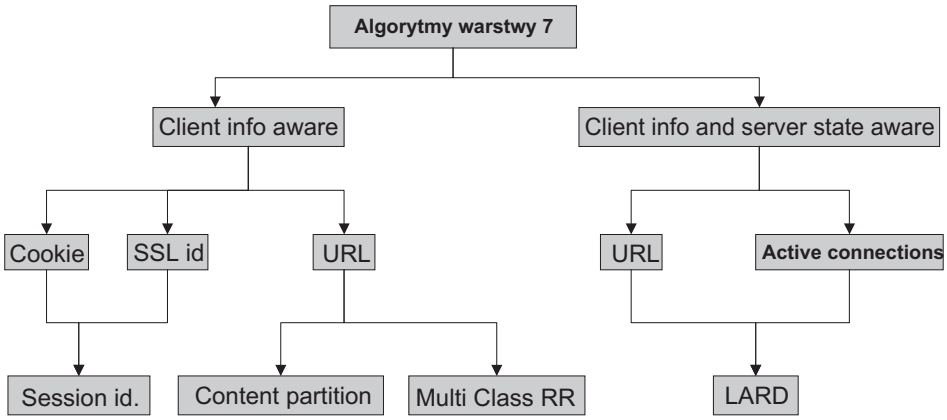
  Ważnym więc elementem funkcjonowania tego rozwiązania jest dobór metryki będącej powyższym parametrem. Istnieje możliwość kontrolowania następujących parametrów:

  - input metric – informacja pozyskiwana jest przez dystrybutor bez współprcy z serwerami, np. liczba aktywnych połączeń miedzy dystrybutorem a poszczególnymi serwerami,
  - server metric – informacja pozyskiwana jest przez serwery i dostarczana dystrybutorowi, np. wykorzystanie procesora lub dysku, czas miedzy otrzymaniem zapytania a wysłaniem odpowiedzi,
  - forward metric – informacja pozyskiwana jest bezpośrednio przez dystrybutor, np. emulacja zapytań do serwerów webowych.

4.3.5 Algorytmy szeregowania wykorzystywane w przełącznikach webowych

Przełączniki webowe mają również możliwość kontrolowania 100% ruchu do serwerów. Zatem i tu konieczne jest wykorzystanie jak najprostszych mechanizmów zarządzania. W przypadku prostych rozwiązań wystarczająco wydajne są algorytmy statyczne. Sytuacja taka występuje, gdy czasy realizacji zleceń przez serwery WWW są bardzo zbliżone do siebie i nie wychodzą poza określony przedział wartości.

W przypadku gdy w systemie występują więcej niż dwa ograniczenia czasu obsługi zlecenia, należy stosować algorytmy dynamiczne wykorzystujące informacje o kliencie lub stanie serwera (client info or server state aware). Mając do dyspozycji heterogeniczne środowisko serwerów trudno jest wybrać najlepszy, wspólny dla wszystkich parametr określający metrykę obciążenia serwera. W takim przypadku preferowane są algorytmy wykorzystujące informacje pochodzące od klientów.



Rysunek 4.2: Algorytmy stosowane w przełącznikach webowych.

Jak wynika z rysunku 4.2, istnieją trzy rodzaje algorytmów opartych na informacjach o kliencie:

- Session identifiers – odwołania HTTP, mające ten sam identyfikator SSL lub ten sam znacznik cookie przypisywane są do tego samego serwera – zmniejsza to czas konieczny na ponowną identyfikację klienta,
- Content partition – podział zasobów serwerów może nastąpić ze względu na:
  - typy plików – dane, pliki graficzne, pliki audio umieszczone są na specjalizowanych serwerach,
  - wielkość plików – duże pliki na szybszych serwerach lub równomierne rozłożenie plików w przypadku serwerów homogenicznych,

- Multi–class round–robin – zasoby są podzielone ze względu na złożoność obliczeniową i czasową, jaka zostanie wygenerowana podczas ich obsługi, np. połączenia szyfrowane wymagają mocy obliczeniowej procesora, odwołania do bazy danych wymagają zwiększonej obsługi dysków, czy wreszcie pobieranie dużych plików w znacznym stopniu obciąża sieć.

Zasada działania algorytmu wykorzystującego informacje o kliencie i serwerze jest następująca. Pierwsze zapytanie klienta o zasoby przekierowywane jest według algorytmu Least loaded server (metryką jest ilość aktywnych połączeń z serwerem). Pozostałe zapytania klienta o ten sam zasób przekierowywane są do tego samego serwera. Dzięki temu zwiększona jest skuteczność odwołań do pamięci podręcznej (cache) tego serwera. Algorytm ten zwany jest Locality-Aware Request Distribution (LARD).

4.3.6 Algorytmy szeregowania wykorzystywane w przypadku przekierowań na poziomie protokołu HTTP

Gdy stosuje się rozwiązania oparte na warstwowej strukturze, istnieje możliwość zarządzania zapytaniami z poziomu protokołu [**?**]. Takie rozwiązanie jest przezroczyste dla użytkowników. Głównym celem stosowania tego mechanizmu jest zapobieganie przeciążeniom serwerów webowych. Przekierowywanie odbywa się poprzez przesłanie klientowi informacji w nagłówku: HTTP OK. 302 – Moved temporary to a new location.

Adres nowej lokalizacji może być podany w postaci nazwy domenowej lub adresu IP. Podanie adresu IP jest bardziej efektywne, ponieważ następuje bezpośrednie odwołanie do nowego serwera (klastra), a nie do serwera DNS. Przekierowania można realizować w zależności od kilku parametrów [**?**]. Proces przekierowań może dotyczyć:

- wszystkich stron,
- tylko stron przekraczających określoną wielkość,
- tylko stron, których ilość obiektów składowych przekracza określoną liczbę,

Wybór serwera, który powinien przejąć zapytanie, może odbywać się z wykorzystaniem jednej z poniższych strategii:

- Round–robin,
- Least Loaded,
- Hash function,
- Client to server proximity.

4.4 Metody równoważenia obciążeń – przykłady

Równoważenie obciążeń z wykorzystaniem filtra datagramów

Inną implementacją rozproszonego algorytmu współdzielenia obciążeń jest zastosowanie na każdym serwerze wchodzącym w skład klastra tzw. filtra datagramów. Klaster taki powinien być połączony z Internetem poprzez pojedynczy router brzegowy. Każdy serwer w klastrze posiada skonfigurowane dwa adresy IP: adres „prywatny" i jednakowy dla wszystkich serwerów adres IP klastra. Router po otrzymaniu datagramu opatrzonego adresem IP klastra nadaje mu fizyczny (sprzętowy np. adres Ethernet) adres typu broadcast (jeżeli do routera przyłączone są tylko serwery należące do klastra) lub multicast (jeżeli klaster jest tylko wyróżnioną grupą wśród wszystkich hostów przyłączonych do serwera). Zastosowanie takiego adresu sprawia, że karty sieciowe wszystkich serwerów w klastrze akceptują datagram. Pomiędzy sterownikiem karty sieciowej, a oprogramowaniem TCP/IP na każdym serwerze musi pracować specjalny proces, który zadecyduje, czy pakiet należy odrzucić czy obsłużyć. Proces ten nazywamy filtrem pakietów [**?**]. Decyzja o odrzuceniu lub obsłużeniu datagramu podejmowana jest na podstawie zawartości dwóch struktur danych: tablicy połączeń TCP (datagramy należące do jednego połączenia TCP muszą być obsługiwane przez serwer który nawiązał dane połączenie) oraz tablicy zawierających wielkość obciążenia poszczególnych serwerów klastra. Tablica ta jest uaktualniana przez same serwery. Każdy serwer musi wysyłać okresowo komunikat typu broadcast zawierający wielkość jego obciążenia. Jeżeli do klastra nadchodzi datagram otwierający nowe połączenie TCP (nagłówek TCP zawiera flagę SYN) serwer o najniższym indeksie obciążenia w tablicy (indeksem tym jest zwykle liczba otwartych połączeń TCP) jest zobowiązany do jego przyjęcia. Dla zwiększenia wydajności klastra w serwerach stosować można dwie karty sieciowe: jedną ze skonfigurowanym adresem IP klastra i drugą z prywatnym adresem serwera. W takim przypadku pakiety, które nie muszą być filtrowane (np. wymiana danych SNMP) przechodzić będą przez „prywatną" kartę. Ten typ równoważenia dotyczyć może każdej usługi korzystającej z TCP, w szczególności WWW. Pierwszą komercyjną implementacją tego rozwiązania był pakiet oprogramowania Convoy Cluster firmy Valence Research przeznaczony dla systemu Microsoft Windows NT. Jako miarę obciążenia serwera przyjęto w nim ilość otwartych połączeń TCP, a komunikaty ogłaszające stan obciążenia wysyłane były przez serwery co sekundę. Oprogramowanie umożliwiało dynamiczną konfigurację klastra przez dodawanie lub usuwanie serwera z klastra bez przerywania pracy klastra. Serwery w klastrze musiały powielać swoje dane. W pierwszej wersji wymagane było stosowanie dwóch kart sieciowych na każdym serwerze, a nadchodzące datagramy rozgłaszane były w trybie broadcast (docierały do każdego hosta w sieci lokalnej klastra, nawet jeśli nie był on serwerem) Wersja 2.0 Convoy Cluster wyeliminowała te niedogodności i oferowała liczne dodatkowe możliwości konfiguracyjne np. opcjonalne stosowanie pokrewieństwa z klientem (ang. *client affinity*), co umożliwia obsługę wszystkich datagramów nadchodzących od raz zidentyfikowanego (w trakcie

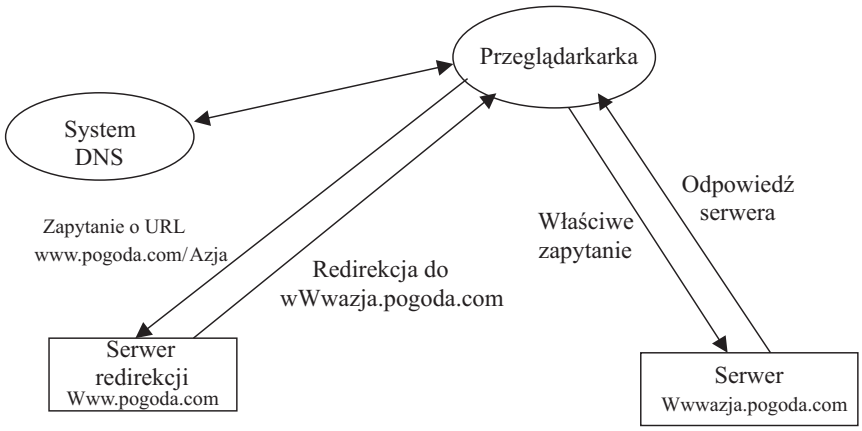
nawiązywania pierwszego połączenia) klienta przez jeden serwer. W roku 1999 firma Microsoft wykupiła od Valence Research technologię Convoy Cluster i po „kosmetycznych" zmianach udostępniła ją w pakiecie Microsoft Windows NT 4.0 Enterprise Server pod nazwą Microsoft Load Balancing Service. Najważniejszą zaletą stosowania filtra pakietów jest jego duża wydajność w porównaniu do scentralizowanych urządzeń rozdzielających zadania (np. LSNAT). Uzyskiwane jest to dzięki temu, że na żadnym etapie obsługi zadania nie są modyfikowane datagramy i nie istnieje pojedynczy punkt podejmowania decyzji o obsłudze zadania. Ważna jest również łatwość konfiguracji klastra i duża niezawodność (serwer, który ulega awarii przestaje wysyłać komunikaty o stanie swego obciążenia, nie jest więc uwzględniany w tablicach obciążenia serwerów w pozostałych węzłach klastra i w ten sposób nie bierze udziału w równoważeniu obciążeń). Koszt jaki należy ponieść, aby uzyskać te niewątpliwie pożądane cechy to trudniejsza konfiguracja poszczególnych serwerów (konieczność instalacji i konfiguracji filtra pakietów) oraz duży ruch w sieci lokalnej klastra wynikający z aktualizacji tablic obciążenia. Aktualizacje te muszą być częste, gdyż łatwo można wyobrazić sobie sytuację, w której serwer o najniższym indeksie obciążenia ulega awarii. W takiej sytuacji pozostałe serwery aż do aktualizacji swoich tablic obciążenia będą odrzucać wszystkie datagramy otwierające nowe połączenia, co z pewnością nie jest zjawiskiem pożądanym.

Równoważenie obciążeń z wykorzystaniem redirekcji HTTP

Redirekcja jest mechanizmem protokołu HTTP, który zaprojektowano z myślą o obsłudze sytuacji w których zasób (plik) wskazywany przez pewien URL zmienia swoje fizyczne położenie (zostaje przeniesiony na inny serwer) i w związku z tym uzyskuje inny URL. Aby nie zmuszać użytkownika do poszukiwania tego zasobu na własną rękę serwer WWW przechowuje tzw. tablice redirekcji. Jest to tablica zawierająca URL, które wcześniej dotyczyły zasobów danego serwera, lecz obecnie zasoby te znajdują się pod innym URL. Tablica zawiera również aktualny URL dla każdego przeniesionego zasobu. W przypadku zapytania o taki „zdezaktualizowany" URL serwer WWW zwraca odpowiedź HTTP typu „przeniesiono" i jako dane przekazuje aktualny URL zasobu. Przeglądarka po otrzymaniu takiej odpowiedzi musi zestawić nowe połączenie z serwerem wskazanym przez otrzymany URL.

Opisany mechanizm w prosty sposób wykorzystać można do pewnego rodzaju równoważenia obciążeń serwerów WWW [**?**, **?**]. W klastrze serwerów wydzielić można tzw. serwer redirekcji, którego nazwa DNS reprezentować będzie cały klaster. Jedynym zadaniem takiego serwera jest utrzymywanie tablicy redirekcji i przekierowanie nadchodzących zapytań do odpowiedniego serwera z klastra. W takiej architekturze serwery zwykle nie powielają swych zasobów, każdy z nich przechowuje część danych udostępnianych przez klaster, a to który z nich obsłuży zapytanie determinowane jest przez rodzaj żądanych informacji. Przykładowo jeśli pod nazwą www.pogoda.com dostępny jest serwis prezentujący prognozę pogody dla każdego kontynentu to zasoby serwisu podzielić można pomiędzy

siedem serwerów (po jednym dla każdego kontynentu) a pod adresem IP stanowiącym odwzorowanie nazwy serwisu należy umieścić serwer redirekcji. W odpowiedzi na zapytanie o dowolny URL zaczynający się np. od ciągu www.pogoda.com/Azja/ serwer redirekcji dokonywałby przekierowania do serwera przechowującego dokumenty o pogodzie w Azji (np. wwwazja.pogoda.com) [**?**]. Poniższy rysunek przedstawia schemat nawiązywania połączenia w przypadku stosowania redirekcji HTTP:



Rysunek 4.3: Schemat redirekcji HTTP.

Stosowanie redirekcji ma zasadniczo dwie zalety. Po pierwsze utrzymanie statycznej tablicy redirekcji jest bardzo proste i tanie w implementacji, nie wymaga stosowania specjalnego sprzętu ani oprogramowania. Po drugie, ponieważ używane są adresy URL, klaster stanowiący logiczną całość może być rozproszony geograficznie tzn. serwer prezentujący dane o pogodzie w Azji może rzeczywiście znajdować się na terenie tego kontynentu, co może być dobrym pomysłem przy założeniu, że o pogodę w Azji pytać będą głównie Azjaci. Redirekcja ma jednak wiele wad [**?**]. Przede wszystkim ograniczona jest do protokołu HTTP, a jak wiadomo wiele łącz hipertekstowych dokonuje przełączenia do np. serwerów FTP, które często pracują na fizycznie tych samych komputerach, co serwery WWW. Druga wada jest wyraźnie widoczna na Rys. 4.3. Aby rozpocząć pobieranie żądanego dokumentu przeglądarka musi dokonać dwóch połączeń, najpierw z serwerem redirekcji, a następnie z właściwym serwerem. Wprowadza to znaczne opóźnienie i powoduje dodatkowe obciążenie sieci, która jest często wąskim gardłem wydajności WWW.

Prezentowane powyżej podejście jest z gruntu statyczne i zakłada wiedzę o tym, które dane będą poszukiwane najczęściej, co umożliwia aprioryczne przydzielenie najsilniejszego serwera w klastrze do obsługi najpopularniejszej części serwisu WWW. Ponieważ tablica redirekcji nie zawiera żadnych danych o obciążeniu poszczególnych serwerów trudno jest dynamicznie uwzględniać takie dane podczas wyboru serwera. W literaturze proponowano architektury piętrowe. Przykładowo dane dotyczące pogody w Azji mogłyby być

powielane pomiędzy kilka serwerów, które raportowałyby stopień swego obciążenia, a na podstawie tych danych serwer redirekcji mógłby dynamicznie aktualizować tablice redirekcji. Zbyt częsta aktualizacja tej tablicy czyni ją jednak bezużyteczną (tablica jest niedostępna w trakcie aktualizacji), a zbyt rzadka powoduje nierównomierność obciążenia. Innym rozwiązaniem jest zachowanie podziału na serwery tematyczne z możliwością dynamicznego przeniesienia części zawartości z serwera mocno obciążonego na serwer posiadający rezerwę wydajności. Aktualizacje tablicy redirekcji byłyby wtedy rzadsze, lecz procedura taka wymagałaby kosztownego śledzenia, które pliki pobierane są najczęściej (tylko przeniesienie takich plików znacząco może zmniejszyć obciążenie serwera), dodatkowo dane byłyby niedostępne przez czas przenoszenia. Obydwie metody dynamicznego wykorzystania tablicy redirekcji mogą być ominięte przez użytkownika, jeśli po połączeniu z serwerem docelowym umieści on zakładkę (ang. *bookmark*) na pobieranych stronach WWW. Statyczna redirekcja HTTP jest skuteczna tylko w przypadku serwisów charakteryzujących się łatwym do przewidzenia wzorcem dostępu do dokumentów.

Równoważenie obciążeń z wykorzystaniem NAT

Mechanizm translacji adresów sieciowych NAT (ang. *Network Address Translation*) został zaprojektowany z myślą o możliwości włączenia prywatnych sieci lokalnych do Internetu z wykorzystaniem jednego, globalnie unikalnego adresu IP dla całej sieci. W standardowej konfiguracji urządzeniem realizującym NAT jest router brzegowy o adresie IP reprezentującym całą sieć, który stanowi jedyne połączenie pomiędzy siecią prywatną a rozległą. Hosty w sieci prywatnej nie muszą posiadać globalnie unikalnych adresów IP, gdyż podczas nawiązywania połączenia z komputerem spoza sieci prywatnej urządzenie NAT zamienia adres nadawcy datagramu IP (pochodzący z sieci prywatnej) na swój własny, dokonuje przeliczenia odpowiednich sum kontrolnych i aby poprawnie kierować datagramami należącymi do jednego połączenia TCP zapamiętuje w wewnętrznych tablicach parametry połączenia (adresy i porty źródłowe i docelowe). Celem wprowadzenia mechanizmu NAT było zapewnienie pewnego stopnia bezpieczeństwa sieciom prywatnym, gdyż jeśli wewnątrz takiej sieci komputery nie posiadają globalnie unikalnych adresów IP, to nie istnieje możliwość nawiązania połączenia z takim komputerem spoza sieci prywatnej. Istnieje wiele rozwiązań komercyjnych realizujących równoważenie (współdzielenie) obciążeń serwerów WWW poprzez mechanizm NAT. Idea polega na kierowaniu zapytań nadchodzących do klastra serwerów poprzez specjalizowane urządzenie NAT, określane czasem jako LSNAT (ang. *Load Sharing NAT*), które kierowałoby zapytanie do konkretnego serwera [**?**]. Algorytm według którego zapytanie byłyby kierowane do serwerów może uwzględniać różnice w ich wydajności jak i stopień ich obciążenia, istnieje również możliwość uwzględnienia w nim numeru portu TCP, co sprawia, że LSNAT stosować można do równoważenia obciążeń wszystkich usług TCP. Obciążenie serwerów określane jest zazwyczaj na podstawie tablicy otwartych połączeń utrzymywanej przez LSNAT dla każdego serwera. Aby dane te były aktualne konieczna jest analiza nagłówków TCP w celu

wykrywania datagramów zamykających połączenie. Serwery w klastrze powinny powielać swoje dane, gdyż wykorzystanie rozproszonego systemu plików powoduje zbyt duże obciążenie sieci lokalnej klastra. Poniżej schematycznie przedstawiono dwie typowe konfiguracje urządzenia NAT jako modułu realizującego równoważenie obciążeń. Na każdym rysunku klaster serwerów reprezentowany jest przez adres IP 172.87.0.100.

W takiej konfiguracji w datagramach przychodzących do klastra następuje zmiana adresu docelowego z adresu urządzenia LSNAT na adres wybranego serwera oraz zmiana adresu nadawcy na adres urządzenia LSNAT. W datagramach wysyłanych w przeciwnym kierunku adres nadawcy zmienia się z adresu serwera na adres LSNAT a adres docelowy z adresu LSNAT na adres rzeczywistego odbiorcy datagramu. Takie postępowanie powoduje, że wszystkie datagramy kierowane do klastra i z powrotem muszą przejść przez urządzenie LSNAT, co umożliwia skonfigurowanie klastra rozproszonego geograficznie. Dzieje się tak kosztem utrzymywania w LSNAT bardziej rozbudowanej (w stosunku do poprzedniej konfiguracji) tablicy połączeń, która umożliwiałaby identyfikację każdego połączenia. Identyfikacji tej dokonuje się wykorzystując numery portów TCP (wraz z translacją adresów datagramu dokonuje się zmiany numeru portu źródłowego na unikalną dla danego serwera wartość powyżej 5000, identyfikacji odpowiedzi adresata dokonuje się na podstawie adresu serwera, który ją wysłał i numeru portu docelowego). Powyższa konfiguracja pracować może również z adresami prywatnymi, uniemożliwia to jednak użycie serwerów odległych geograficznie. Poniżej przedstawiono krótki opis dwóch popularnych rozwiązań komercyjnych wykorzystujących mechanizm NAT do równoważenia obciążeń serwerów WWW.

Rozwiązania korzystające z mechanizmu NAT do równoważenia obciążeń są znacznym postępem w stosunku do metod opisanych wcześniej w tej pacy. Umożliwiają skuteczne uwzględnienie stopnia obciążenia poszczególnych serwerów w klastrze jak i ich indywidualnych właściwości. LSNAT umożliwia rozróżnianie połączeń na podstawie numerów portów TCP jak i równoważenie obciążeń pomiędzy serwery odległe geograficznie. Metoda ta nie jest jednak pozbawiona wad. W oczywisty sposób urządzenie LSNAT staje się wąskim gardłem wydajności klastra, gdyż cały ruch pomiędzy klasterem, a Internetem musi przez nie przechodzić. Jeśli wziąć pod uwagę, że w przypadku WWW objętość odpowiedzi serwera jest co najmniej dziesięciokrotnie większa niż zapytanie, jasnym staje się, że w obliczu ciągłego wzrostu liczby użytkowników, najwydajniejsze nawet urządzenie LSNAT może w końcu ograniczyć wydajność klastra. Należy również zauważyć, że zmiana adresu IP w nagłówku datagramu pociąga za sobą konieczność wyliczenia nowej sumy kontrolnej dla całego datagramu. Jest to operacja czasochłonna przez co wprowadza opóźnienie w transmisji danych jak i konieczność kolejkowania pakietów w samym urządzeniu (trudno oczekiwać, że nawet urządzenie przetwarzające klika pakietów równolegle poradzi sobie bez opóźnień z całym przechodzącym przez nie ruchem). Ta właściwość LSNAT znacznie ogranicza jego skalowalność, gdyż dodawanie nowych serwerów do klastra w prosty sposób zwiększa kolejkę pakietów w urządzeniu, aż do momentu, w którym przekroczone zostaną jego możliwości lub cierpliwość użytkowników.

Równoważenie obciążeń poprzez „pół–połączeniowe" marszrutowanie TCP

Rozpatrując przedstawione kolejno w tej pracy metody równoważenia obciążeń można zauważyć pewną prawidłowość. Otóż im dana metoda jest bardziej skuteczna i zaawansowana koncepcyjnie, tym w niższej warstwie sieciowej operuje. Redirekcja HTTP i RR–DNS działały powyżej warstwy transportowej, w ogóle nie ingerując w zawartość datagramów IP. Rozwiązania oparte o LSNAT i DPR pracowały w warstwie sieciowej i aby skutecznie rozdzielać zadania pomiędzy serwery musiały dokonywać modyfikacji (adresów i sum kontrolnych) w nagłówkach datagramów IP. Metoda opisana w tym punkcie operuje w warstwie fizycznej i do rozdziału zadań nie musi zmieniać zawartości datagramów IP.

„Pół–połączeniowe" marszrutowanie TCP (ang. *half–connection TCP routing*) jest opatentowaną przez firmę IBM technologią, która legła u podstaw zasady działania pakietu oprogramowania Network Dispatcher [**?**, **?**]. W swej podstawowej konfiguracji pakiet ten umożliwia zestawienie klastra złożonego z połączonych siecią lokalną serwerów korzystających TCP lub UDP (w szczególności serwerów WWW) i udostępnienie go pod jednym adresem IP. W klastrze tym serwery mają unikalne globalnie lub lokalnie adresy IP i muszą powielać swoje dane. W obszarze tej samej sieci lokalnej, w której działa klaster, musi być wyznaczony komputer, na którym pracować będzie oprogramowanie Network Dispatcher. Komputer ten musi posiadać dwa adresy IP, jeden z nich, tzw. adres NFA (ang. *Non–Forwarding Address*) jest „osobistym" adresem komputera, pod którym można skontaktować się z pracującym na tym komputerze oprogramowaniem (np. z agentem SNMP). Drugi adres, to adres reprezentujący klaster serwerów, wszystkie datagramy IP opatrzone tym adresem będą przetwarzane przez program Network Dispatcher i na podstawie algorytmu współdzielenia obciążeń przekazywane do jednego z serwerów w klastrze. Dispatcher zmienia jedynie docelowy adres fizyczny (sprzętowy) datagramu, zawarty w nagłówku sprzętowym, dodawanym przed nagłówek IP (np. w nagłówku Ethernet). Dzięki temu datagramy wysyłane przez serwer w odpowiedzi mogą być kierowane bezpośrednio do klienta, bez konieczności ponownego przejścia przez oprogramowanie Network Dispatcher (w celu np. przywrócenia oryginalnych adresów IP). Stąd pochodzi nazwa tej metody – „pół–połączeniowe" marszrutowanie TCP. Serwer wysyłając odpowiedź dokonuje standardowej zamiany adresów IP nadawcy i odbiorcy pobierając obydwa te adresy z nagłówka otrzymanego datagramu. Jak pamiętamy adresem docelowym jest w tym datagramie adres klastra (komputera, na którym pracuje Dispatcher), więc adres ten stanie się adresem nadawcy odpowiedzi, co sprawi, że kolejne datagramy dotyczące danego połączenia skierowane zostaną na adres Network Dispatcher–a [**?**].

Adres IP każdego serwera w klastrze jest różny od adresu klastra. Fakt, że pomimo to oprogramowanie TCP/IP w serwerze akceptuje pakiety opatrzone adresem klastra jest możliwy dzięki dodaniu aliasu do adresu interfejsu pętli zwrotnej (ang. *loopback interface*) w każdym serwerze. Do standardowego adresu 127.0.0.1 dodawany jest jako alias adres klastra (w przypadku przedstawionym na rysunku 139.37.38.39). Możliwość nadawania wielu adresów interfejsowi pętli zwrotnej jest jedynym wymaganiem Dipatcher-a w sto-

sunku do serwerów. Ponieważ Network Dispatcher rozróżnia porty TCP i UDP możliwe jest równoważenie obciążeń powodowanych przez dowolny protokół korzystający z TCP lub UDP m.in. HTTP (WWW), FTP, SSL, SMTP, POP3 czy Telnet. Ponieważ wszystkie serwery powielają swoje dane, możliwa jest sytuacja w której np. plik HTML opisujący stronę WWW pobierany jest z serwera A, a pliki graficzne składające się na stronę pobierane są z serwera B. W celu zarządzania połączeniami Network Dispatcher przechowuje dwie struktury danych- tablicę połączeń aktywnych (otwartych) i tablicę nowo przydzielonych połączeń. Tablica otwartych połączeń służy do poprawnego kierowania datagramów dotyczących nawiązanego połączenia TCP (połączenie TCP nie może być przez Dipatcher przekazane pomiędzy serwerami). Zawiera ono adres IP nadawcy i numer źródłowego portu TCP oraz adres IP serwera, który obsługuje połączenie i numer portu docelowego TCP oraz pole stanu połączenia. Pozycje z tej tablicy są usuwane po wykryciu w nagłówku TCP flagi FIN lub RST. Ilość połączeń otwartych na danym serwerze jest również uwzględniana podczas obliczenia wagi serwera. Tablica nowo przydzielonych połączeń służy do zapamiętania jak wiele połączeń zostało przydzielonych do danego serwera od ostatniego odświeżenia wag serwerów i jako miara prędkości zmian obciążenia serwera wykorzystywana jest do obliczenia jego wagi.

Jeżeli na adres Dispatcher–a przychodzi datagram nie należący do żadnego z połączeń zapisanych w tablicy aktywnych połączeń, oznacza to, że jest to nowe połączenie i należy przydzielić serwer do jego obsługi. Dispatcher utrzymuje cykliczną listę serwerów zawierającą ich wagi. Waga serwera oddaje stopień jego obciążenia i jest obliczana przez Dispatcher okresowo dla klastra (dla wszystkich serwerów jednocześnie). Dispatcher pamięta numer i wagę serwera do którego przydzielono ostatnie połączenie TCP i rozpoczynając przeszukiwanie listy od tego miejsca poszukuje serwera o wadze większej lub równej zapamiętanej wadze (im większa waga, tym mniej obciążony serwer). Do znalezionego w ten sposób serwera przydzielane jest nowe połączenie, co znajduje odwzorowanie w tablicy aktywnych połączeń.

Waga dla każdego serwera obliczana jest na podstawie ilości otwartych połączeń TCP, ilości nowo przydzielonych połączeń, stopnia obciążenia procesora w serwerze (miara ta uwzględnia również obciążenie wynikające z zadań uruchamianych lokalnie i pochodzi od modułu ISS (ang. *Interactive Session Support*) pakietu Network Dispatcher, ISS uwzględnia również indywidualne właściwości serwera) oraz na podstawie danych pochodzących z tzw. Advisor–ów. Advisor jest programem symulującym klienta danego protokołu i bada jak szybko serwer jest w stanie zareagować na typowe dla danego protokołu żądanie. Np. Advisor HTTP wysyła do serwera żądanie HTTP GET / (prześlij domyślny dokument głównego katalogu w serwisie) i jako wynik zwraca czas, po którym otrzymał pierwszy bajt odpowiedzi. Proporcje z jakimi poszczególne elementy wchodzą w skład wagi, jak i częstotliwość odświeżania wag ustalane są przez administratora klastra. Network Dispatcher posiada wiele cech wyróżniających go spośród przedstawionych wcześniej rozwiązań. Jest oczywiście wolny od niekorzystnych cech RR–DNS i nie modyfikuje datagramów IP, a analiza, którą na nich prowadzi (zapamiętanie adresów i portów, wykrywanie flag), nie

jest czasochłonna. Dodatkowo ruch przechodzący przez Dispatcher–a stanowią tylko datagramy nadchodzące do klastra (typowo mniejsze od odpowiedzi), dzięki czemu nie jest łatwo (w przeciwieństwie do urządzeń LSNAT) tak obciążyć Dispatcher–a by stał się „wąskim gardłem" wydajności klastra. W przeciwieństwie do rozwiązania z zastosowaniem DPR, Dispatcher nie wymaga specjalistycznego oprogramowania pracującego na serwerze, więc nie obciąża go np. zadaniem przetwarzania datagramów IP.

Pierwszy prototyp Network Dispatcher–a obsługiwał Internetowy serwis IO w Atlancie w 1996. Kolejne, już komercyjne wersje obsługiwały takie wydarzenia jak turniej US Open i IO w Nagano w 1998 oraz mecze szachowe Deep Blue vs. Garri Kasparow. Szczytowe obciążenie klastra w przypadku IO w Nagano osiągnęło 110 414 zapytań na minutę, a mimo to równoważenie obciążeń przy użyciu Network Dispatcher–a zapewniło wszystkim użytkownikom dobry czas odpowiedzi i zadowalający transfer.

Rówoważenie obciążeń poprzez bezpośrednie routowanie – *Direct Routing*

Architektura ta jest podobna do wykorzystywanej w produkcie firmy IBM – oprogramowaniu SecureWay Network Dispatcher.

Adres wirtualny serwera jest współdzielony poprzez poszczególne nody i load balancer. Interfejs sieciowy dystrybutora jest skonfigurowany także do wirtualnego adresu, który jest wykorzystywany do przyjmowania pakietów przychodzących oraz do bezpośredniego routowania pakietów do wybranych serwerów. Wszystkie rzeczywiste serwery mają swoje non–arp aliasy interfejsu sieciowego skonfigurowane z adresem wirtualnym lub bezpośrednio przekierowują pakiety przeznaczone na adres wirtualny do lokalnych gniazd, w ten sposób, że rzeczywiste serwery mogą przenosić pakiety tylko lokalnie. Zarówno load balancer jak i rzeczywiste serwery muszą mieć interfejsy sieciowe fizycznie skojarzone z HUB–em lub switchem. Wygląda to w ten sposób, że dystrybutor po prostu zmienia adres MAC na adres rzeczywistego serwera i retransmituje do niego pakiet.

4.5 Przykłady produktów stosowanych do równoważenia obciążenia wielokomputerowych serwerów WWW

4.5.1 LinuxVirtualServer

Linux Virtual Server jest wysoce skalowalnym i dostępnym serwerem zbudowanym na klastrze rzeczywistych serwerów wraz z możliwością realizacji równoważenia obciążęń. System ten oparty jest na systemie Linux. Architektura tego rozwiązania jest (jak i pozostałe) przezroczysta dla klienta i odbywa się na poziomie protokołu IP (warstwa czwarta).

Rzeczywiste serwery mogą być połączone w obrębie sieci lokalnej lub geograficznie rozproszone w sieci WAN. Fron-endem tych rzeczywistych serwerów jest dystrybutor

(load balancer), który marszrutuje żądania do różnych serwerów oraz powoduje, że równoległe usługi działające w obrębie klastra wydają się być jedną wirtualną usługą dla całego klastra na jednym adresie IP. Skalowalność w tym systemie oznacza, że w przezroczysty sposób można dodawać i usuwać poszczególne nody do klastra. Wysoka dostępność jest realizowana poprzez detekcję uszkodzonych nodów lub niesprawnych demonów oraz równoczesną rekonfigurację systemu.

Virtual Server można implementować na trzy sposoby:

Virtual Server poprzez NAT – zaletą tego rozwiązania jest fakt, że rzeczywiste serwery mogą pracować na dowolnym systemie operacyjnym, który włada protokołem TCP/IP. Rzeczywiste serwery mają prywatny adresy IP i tylko one są potrzebne dystrybutorowi do pracy. Wadą tego rozwiązania jest raczej niewielka skalowalność, ponieważ w tym wypadku *load balancer* może stanowić wąskie gardło całego systemu gdy liczba podłączonych serwerów będzie wynosić około 20 lub więcej. Jest to spowodowane tym, że zarówno ruch przychodzący (niewielki) jak i wychodzący (o wiele większy) są przepisywane przez dystrybutora. Można to ominąc poprzez korzystanie z pozostałych rozwiązań Virtual Servera lub poprzez rozwiązanie hybrydowe z DNS i kilkoma osobnymi Virtual Serverami;

Virtual Server poprzez Tunelowanie IP – w tym wypadku load balancer tylko przekazuje ruch wchodzący do poszczególnych rzeczywistych serwerów, zaś one odpowiadają bezpośrednio do użytkowników. W tym rozwiązaniu jak widać serwer virtualny może się składać i z ponad 100 serwerów i nadal dystrybutor nie będzie stanowił wąskiego gardła. Maksymalna wydajność Virtual Servera w tym przypadku może sięgać powyżej 1Gbps – w przypadku gdy dystrybutor dysponować będzie 100Mbps kartą sieciową. Rozwiązanie oparte na tunelowaniu IP może być używane do serwerów wirtualnych w bardzo wysokich wydajnościach, szczególnie dobrych do tworzenia virtualnych proxy serwerów. Wadą tego rozwiązania jest to, że każdy serwer musi umieć dokonywać enkapsulacji IP (tunelowanie IP) wewnątrz IP;

Virtual Server poprzez bezpośrednie routowanie [2] – opisany powyżej. Wadą tego rozwiązanie jest brak możliwości rozbudowy wirtualnego serwera powyżej sieci lokalnej, jednakże w porównaniu z poprzednią architekturą rzeczywiste serwery nie potrzebują posługiwać się enkapsulacją IP.

W połączeniu z każdą implementacją Virtual Server korzysta z następujących algorytmów dystrubuujących pakiety:

- marszrutowanie algorytmem Round–Robin;
- marszrutowanie algorytmem Weighted Round–Robin (statyczne wagi, bez wykorzystywania informacji o stanie systemów);

[2] ang. *Direct Routing*

- marszrutowanie typu Least Connection (dynamiczny algorytm – opisany w tekscie);
- marszrutowanie typu Weighted Least Connection (jak wyżej + statycznie nadawane wagi poszczególnym serwerom);
- marszrutowanie Locality–Based Least Connection;
- marszrutowanie Locality–Based Least Connection witch Replcation;
- marszrutowanie typu Destination Hashing;
- marszrutowanie typu Source Hashing.

Zaletą Virtual Servera jest jego działanie w obrębie jądra co oznacza wysoką wydajność i stabilność. Kolejną zaletą jest dostępność kodu źródłowego (oprogramowanie typu OpenSource) co oznacza możliwość modyfikacji kodu w zależności od potrzeb.

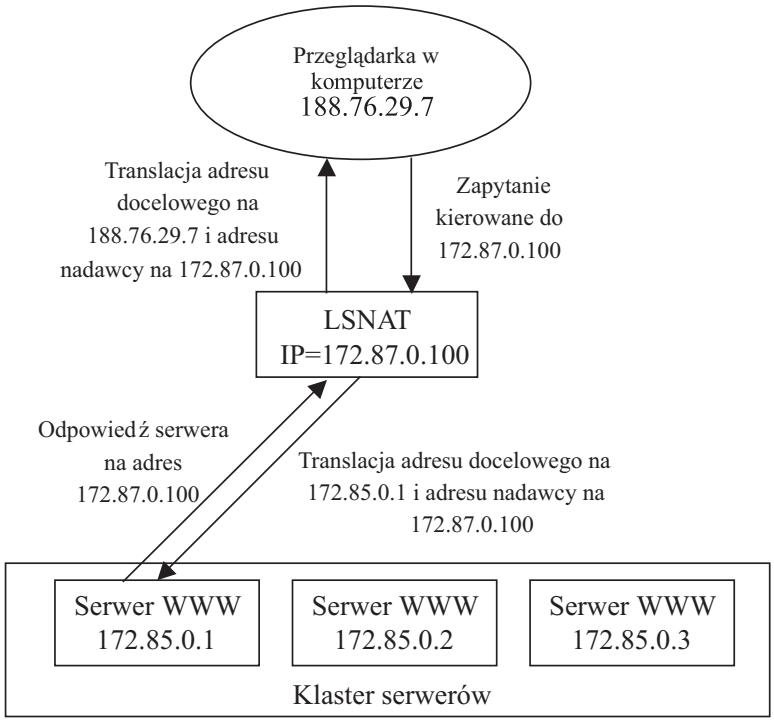
4.5.2 Cisco LocalDirector

LocalDirector jest nazwą rodziny urządzeń produkowanych przez firmę Cisco. Urządzenia te służą do równoważenia obciążeń dowolnych serwerów wykorzystujących TCP (np. serwery WWW, FTP, SSL) [**?**, **?**]. LocalDirector instalowany jest w konfiguracji przedstawionej na Rys. 4.4 z wykorzystaniem adresów prywatnych,

musi więc być instalowany jako jedyne połączenie pomiędzy klasterem a bramką (ang. *gateway*) do sieci rozległej. LocalDirector wymaga powielania zawartości pomiędzy serwerami w klastrze. Serwery mogą być połączone z LocalDirector-em poprzez sieć Ethernet, FastEthernet lub FDDI. Wybór serwera dokonywany jest sekwencyjnie na podstawie wag uwzględniających indywidualne właściwości serwera i jego obciążenie w postaci liczby otwartych połączeń TCP. LocalDirector sprawdza czy serwer się nie załamał poprzez okresowe nawiązywanie z nim połączenia kontrolnego (Ping, HTTP GET /). Producent zapewnia, że LocalDirector, w zależności od modelu, jest w stanie obsłużyć od 7000 do 25000 połączeń na sekundę przy przepustowości od 80 Mbit/sek. do 400 Mbit/sek. Istnieje możliwość piętrowej konfiguracji LocalDirector-a. Kilka rozproszonych geograficznie klasterów, z których każdy obsługiwany jest przez jedno urządzenie LocalDirector, można zaprezentować jako jeden klaster z użyciem tzw. GlobalDirectora. GlobalDirector to w rzeczywistości serwer DNS rozdzielający zapytania pomiędzy klastery na zasadzie podobnej do RR–DNS.

4.5.3 F5 Labs BigIP

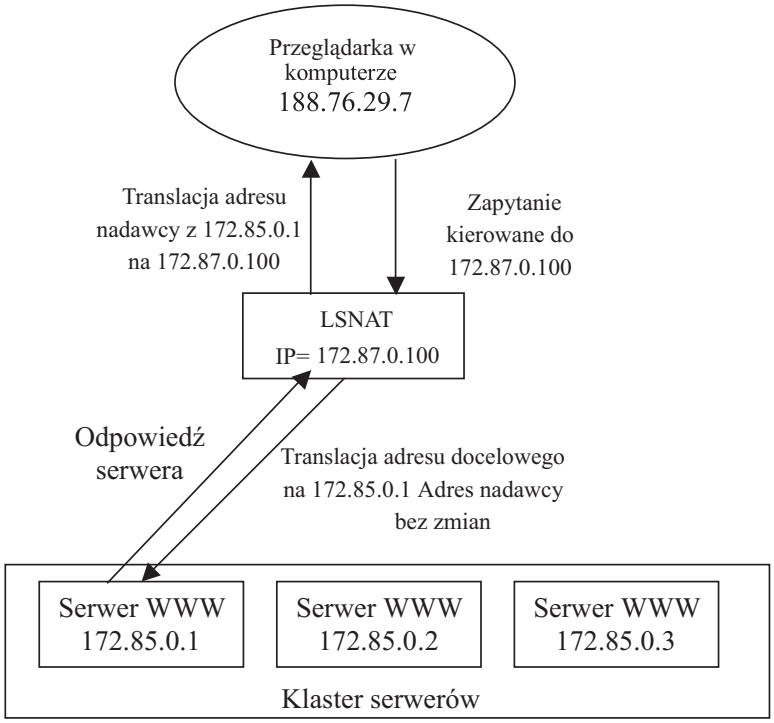
BigIP firmy F5 Labs jest tzw. rozwiązaniem „pod klucz” (ang. *turn–key solution*). Oznacza to, że dostarczany jest tak jak urządzenie sprzętowe (np. LocalDirector), choć w rzeczywistości jest to komputer pracujący pod kontrolą specjalizowanego oprogramowania i



Rysunek 4.4: LSNAT symetrycznie zmieniający adres IP.

systemu operacyjnego firmy F5 Labs. BigIP umożliwia równoważenie obciążeń każdego serwera korzystającego z protokołu TCP lub UDP [**?**, **?**]. Urządzenie może być podłączone w konfiguracji przedstawionej na Rys. 4.4 lub Rys. 4.5 w obydwu jednak przypadkach, aby uniemożliwić ominięcie

mechanizmu równoważenia obciążeń, BigIP powinien znajdować się pomiędzy klasterem serwerów, a bramką do Intetrnet-u. BigIP można połączyć z klasterem poprzez sieć Ethernet, FastEthernet, FDDI lub opcjonalnie GigaBitEthernet. Urządzenie oferuje do wyboru siedem algorytmów rozdziału zadań. Trzy z nich to algorytmy statyczne, są to: algorytm cykliczny (ang. *Round–robin*); algorytm proporcjonalny przydzielający zadania według ustalonych na stałe wag i algorytm priorytetowy, który umożliwia wydzielenie w klastrze grup serwerów o określonym priorytecie i rozdział zadań do grup, w każdej z grup serwer do obsługi konkretnego zadania wskazywany jest cyklicznie. Pozostałe algorytmy są dynamiczne i uwzględniają obciążenie poszczególnych serwerów. Są to algorytm LC (ang. *Least Connections*) przydzielający zadania do serwera utrzymującego najmniej otwartych połączeń, algorytm wyznaczający ten serwer, który najszybciej odpowie na zapytanie kontrolne (np. HTTP GET /), tzw. algorytm obserwacyjny będący kombinacją powyższych i algorytm predyktywny, który przydziela zadania (połączenia) do serwera, którego obciążenie (mierzone jako kombinacja ilości otwartych połączeń i czasu odpowiedzi na zapytanie kontrolne) zmniejszało się najszybciej w ciągu np. ostatnich 5 sekund.



Rysunek 4.5: LSNAT zmieniający jeden z adresów IP

Konkretne modele BigIP wyposażone są w procesory Intel Pentium II 450 do 600 MHz pamięć RAM 128 MB do 1GB i dysk twardy 4 GB do 8.4 GB. Producent gwarantuje przepustowość od 170 Mbit/sek. do 350 Mbit/sek. i obsługę do 20000 zapytań na sekundę.

4.5.4 IBM SecureWay Network Dispatcher

W związku z tym, że pakiet ten jest jednym z głównych elementów tej pracy, jego dokładny opis i konfiguracja znajduje się w następnym rozdziale (Rozdz. 5).

Rozdział 5

System do zarządzania wielokomputerowym serwerem WWW

5.1 Wstęp

W tym rozdziale zostanie przedstawiony opis konfiguracji stanowiska użytego do badań nad charakterystykami różnych konfiguracji i algorytmów klastra serwerów WWW zestawionego w środowisku systemu AIX i Windows NT z wykorzystaniem pakietu IBM WebSphere Performance Pack. Opisana zostanie konfiguracja sieci komputerowej, w której zostały przeprowadzone badania oraz charakterystyka poszczególnych testów.

5.2 Charakterystyka użytego oprogramowania

5.2.1 IBM WebSphere Performance Pack

IBM WebSphere Performance Pack jest oprogramowaniem infrastrukturalnym WWW wiążącym ze sobą skalowalność, niezawodność i wydajność, które to cechy są niezbędne dla aplikacji e-biznesu zarówno w środowiskach lokalnych jak i rozproszonych. Jego funkcje łączą ze sobą znakomity caching, zarządzanie plikami i równoważenie obciążenia, które razem kompensują wrodzone słabości Internetu by wspierać krytyczne aplikacje biznesowe.

IBM WebSphere Performance Pack składa się z trzech głównych elementów, które to pozwalają zredukować obciążenie serwera WWW, zwiększyć dyspozycyjność zasobów (zawartości) i zwiększyć wydajność serwera WWW [**?**]:

Współdzielenie plików

Komponent zajmujący się współdzieleniem plików, znany jako IBM AFS Enterprise File System (AFS), jest systemem plików pozwalającym współpracującym hostom

(klientom i serwerom) efektywnie współdzielić zasoby systemów plików poprzez zarówno LAN jak i WAN. Prowadzi on replikację informacji pomiędzy wieloma serwerami w czasie rzeczywistym, gwarantując przy tym spójność danych, dostępność, stabilność i efektywność w administrowaniu, wymaganych przez duże, rozproszone serwisy webowe.

Keszowanie i filtrowanie

Komponent odpowiedzialny za pamięć podręczną i filtrowanie zawartości webowej, znany jako IBM Web Traffic Express (WTE) jest proxy serwerem, który dostarcza wysoce skalowalnych funkcji keszowania i filtrowania związanych z przesyłaniem żądań webowych i dostarczaniem adresów URL. Moduł ten jest w stanie zredukować kosztowną szerokość wykorzystywanego pasma dostępowego i w szybszy sposób, oraz z mniejszymi opóźnieniami dostarczać informacje do klienta.

Równoważnie obciążenia

Moduł odpowiedzialny za równoważenie obciążeń, znany jako IBM SecureWay Network Dispatcher jest serwerem zdolnym do dynamicznego monitorowania i równoważenia aplikacji i serwerów TCP w czasie rzeczywistym. Główną zaletą tego komponentu jest możliwość dynamicznego związania ze sobą wielu serwerów TCP tak, że wyglądają z sieci jak pojedynczy logicznie serwer.

Każdy z elementów IBM WebSphere Performance Pack może być zainstalowany oddzialenie od pozostałych – także na innych komputerach. Dzięki związaniu tylu elementów w jednym pakiecie - klient otrzymuje oprogramowanie o scentralizowanej administracji i zminimalizowanym koszcie.

Procedury instalacyjne pozwalają wybrać, który komponent należy zainstalować i na jakich maszynach w sieci mają się one znajdować. Oprogramowanie to jest portowane na następujące platformy: AIX (od wersji 4.2.1), Solaris (od wersji 2.6) oraz MS Windows NT i 2000.

5.2.2 Web Traffic Express

WTE jest naraz kaszującym serwerem proxy i filtrem zawartości pakietów. Zaawansowane keszowanie pozwala zminimalizować wykorzystanie przepustowości zwiększając przy tym pewność, że klienci spędzą znacznie mniej czasu podczas pobrania tej samej informacji kilka razy [**?**, **?**].

Tradycyjny proxy serwer przesyła żądania dla URL od klienta i podaje je dalej do serwera przeznaczenia. WTE daje coś więcej; pozwala zapisać lub keszować dokumenty, które przesyła oraz serwować je podczas późniejszych żądań ze swojego keszu, a nie ze źródłowego serwera. Co oznacza, że klient żądany dokument otrzymuje szybciej przy zmniejszonym obciążeniu łącz.

Moduł ten posiada także dodatkowe cechy takie jak:

- możliwość utrzymania bardzo dużego keszu;
- opcję automatycznego odświeżania tej części pamięci podręcznej zawierającej najczęściej popobierane strony;
- możliwość keszowania także tych stron, których nagłówek wymaga by były zawsze pobierane ze źródłowego serwera;
- konfigurowania okresowego porządkowania keszu z informacji bezużytecznych w celu poprawy wydajności i utrzymania efektywności jego działania;
- Remote Cache Access (RCA) - właściwość pozwalająca na wielu maszynom z WTE uwspólniania tego samego keszu poprzez rozproszony system plików, taki jak AFS, w celu zredukowania redundancji zawartości.

Dodatkowo WTE pozwala na ustawianie filtrowania zawartości na poziomie serwera proxy – pozwalając na takie zabiegi jak np. blokowanie URL–ów. Zwielokrotnione serwery WTE mogą być obciążeniowo zrównoważone.

Kolejną cechą jaką posiada ten moduł jest możliwość pracy jako przezroczysty proxy – czyli taki, do którego działania nie jest potrzebna żadna zmiana w konfiguracji przeglądarki klienta – WTE pracuje wtedy na porcie serwera WWW.

5.2.3 IBM SecureWay Network Dispatcher

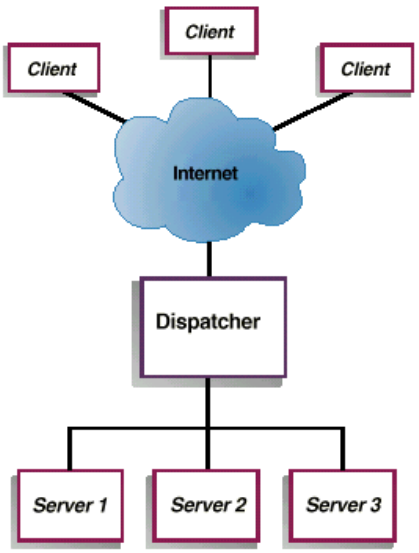
Jest to jedno z pierwszych komercyjnych rozwiązań, które pozwala zwiększyć wydajność i zapewnić ciągłą pracę serwisów internetowych. Znalazło ono już szerokie zastosowanie w znaczących serwisach WWW, takich jak serwisy prowadzone w trakcie olimpiad w Atlancie i Nagano. System jest przeznaczony do obsługi serwerów webowych zbudowanych w technologii klastrowej. Klaster jest grupą serwerów webowych obsługujących jedną witrynę WWW. Wszystkie serwery w klastrze posiadają tę samą zawartość i zapytanie może być obsłużone przez dowolny z nich serwer. Pojedyncze zapytanie obsługiwane jest przez jeden z aktualnie sprawnych serwerów. IBM SecureWay Network Dispatcher składa się z trzech komponentów: podsystemu Dispatcher, podsystemu Interactive Session Support (ISS) oraz podsystemu Content–based Routing (CBR). Komponenty te mogą być używane łącznie lub każdy z osobna [**?**].

Komponent Dispatcher

Podstawowym komponentem systemu jest Dispatcher, którego zadaniem jest zapewnienie równomiernej dystrybucji zapytań realizowanych za pośrednictwem protokołu TCP/IP

między wieloma serwerami realizującymi tę samą usługę w konfiguracji klastrowej. Funkcjonowanie Network Dispatcher'a opiera się na architekturze przedstawionego wcześniej dystrybutora. Dispatcher może równoważyć obciążenia w obrębie sieci lokalnej lub rozległej. Dla każdego klastra definiuje się porty, które chcemy, aby były obsługiwane przez klaster, następnie serwery, które będą dostarczać usługi na każdym z zdefiniowanych portów. Dispatcher może obsługiwać wiele klastrów [**?**].

Wysoką dostępność serwisu osiągnięto poprzez działanie samego Dispatcher'a, który wykrywa niesprawne serwery w klastrze i omija je przy dystrybucji zapytań oraz poprzez wprowadzenie drugiego systemu Dispatcher'a. W podstawowym trybie pracy Dispatcher wymaga, aby wszystkie serwery w klastrze były w tej samej podsieci co Dispatcher. Wówczas, aby podwyższyć dostępność serwisu, możemy skonfigurować drugi system Dispatcher'a, który pełnić będzie rolę maszyny zapasowej i czuwającej w gotowości do przejęcia zadania równoważenia obciążenia w przypadku, gdyby maszyna podstawowa Dispatcher'a przestała działać poprawnie. Mechanizm typu heartbeat zapewnia monitorowanie stanów przez obie maszyny, podstawową i zapasową oraz przejęcie zadań w przypadku wystąpienia awarii.



Rysunek 5.1: Konfiguracja logiczna modułu dispatcher

Dispatcher realizuje swoje zadania za pomocą trzech wewnętrznych komponentów: Egzekutora, Menadżera i Doradców. Egzekutor realizuje równoważenie obciążeń. Dla pakietu wysyłanego w ramach nowego połączenia między klientem a serwerem WEB Dispatcher sprawdza, który z serwerów może przejąć obsługę zlecenia na żądanym przez klienta porcie i adresie klastra. Następnie, dla każdego takiego serwera, na podstawie zgromadzonych wag określających poziom obciążeń serwerów, określa serwer najmniej obciążony,

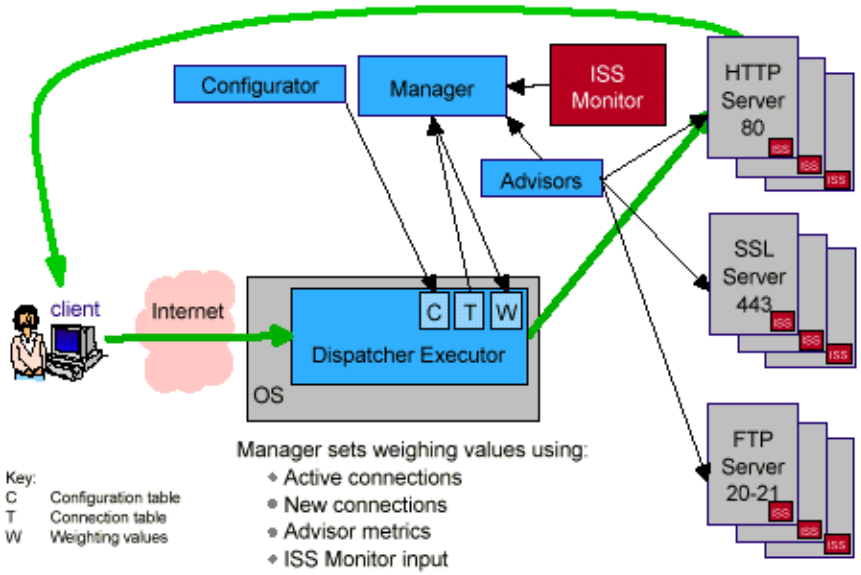
do którego przekazuje pakiet. Jeśli połączenie już istnieje, wtedy bez żadnego przetwarzania pakiet jest natychmiast wysyłany do tego samego serwera, który został wybrany podczas inicjalizacji połączenia. Rozdział zleceń między serwery bazuje na wartościach wag wskazujących na możliwość potencjalnego obciążenia serwera – np. jeżeli jeden serwer ma wagę 2, a drugi ma wagę 1, to serwer o wadze 2 powinien dostać dwa razy więcej żądań niż ten o wadze 1. Wagi mogą być ustawiane ręcznie lub przez Menadżera. Najczęściej do ustawiania wag korzysta się z Menadżera, który ustawia wagi automatycznie i w sposób adaptacyjny, z uwzględnieniem aktualnych warunków pracy klastra. Menadżer może korzystać z wewnętrznych liczników systemowych oraz z informacji dostarczanych przez inne komponenty systemu, takich jak ISS czy WLM. Zarządca decyduje który z serwerów jest najmniej obciążonym poprzez obserwację wag serwerów, które to okresowo analizuje i uaktualnia. Decyzję jaką serwerowi nadać wagę podejmuje opierając się na czterech parametrach:

- liczba aktywnych połączeń realizowanych na każdym serwerze TCP;
- liczba nowych połączeń na każdym serwerze TCP;
- danych wejściowych pochodzących od doradców;
- informacji pochodzących od narzędzi monitorujących pracę systemu, takich jak ISS;

Używanie zarządcy jest opcjonalne, ale jeśli nie jest on używany, równoważenie obciążenia jest dokonywane przy użyciu marszrutowania algorytmem ważonego Round Robina, gdzie wagi poszczególnych serwerów WWW są nadawane statycznie;

Aby dać administratorowi większą kontrolę nad tym, gdzie kierowane będą żądania różnego typu pochodzące od różnych grup użytkowników, wprowadzono pojęcie reguł i grup serwerów im podległych, tj. serwerów, wśród których będzie realizowane równoważenie obciążenia w razie spełnienia reguły. Dla Dispatcher'a dostępne są reguły bazujące na: adresie IP klienta, porcie klienta, porze dnia, połączeniach na sekundę dla danego portu, aktywnych połączeniach dla danego portu. Reguły dają możliwość implementacji wysokiej jakości usług dla wybranych klientów bądź w określonej porze dnia. Dostępna jest także reguła „zawsze prawdziwe", która oznacza spełnienie zadanego warunku dla wszystkich przyjmowanych zleceń.

Zadaniem doradców jest zbieranie informacji na temat obciążenia serwerów. Doradcy sprawdzają również, czy serwery, na których dokonywane jest równoważenie obciążeń, funkcjonują prawidłowo. Doradcy okresowo otwierają połączenia TCP i wysyłają do serwera wiadomość z żądaniem specyficznym dla danego typu doradcy. Po wysłaniu wiadomości Doradcy czekają na odpowiedź. Po jej otrzymaniu większość Doradców szacuje wartość obciążenia serwera na podstawie czasu, jaki upłynął nim serwer zwrócił odpowiedź na żądanie i zgłaszają ten czas (w milisekundach) Menadżerowi, aby ten na podstawie tej i innych informacji oszacował wartość wag poszczególnych serwerów.



Rysunek 5.2: Architektura Network Dispatcher-a

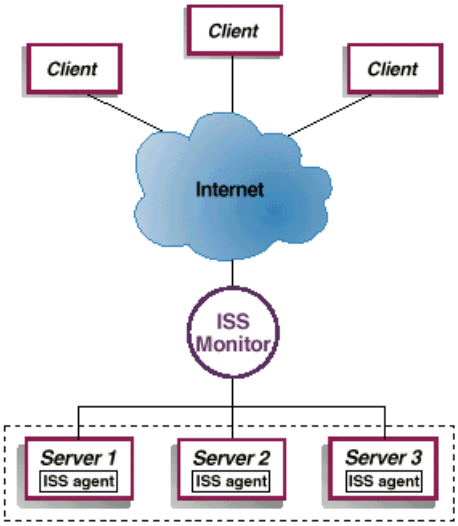
Wraz z produktem IBM SecureWay Network Dispatcher dostarczeni są doradcy: HTTP, FTP, Telnet, NNTP, POP3, SMTP, SSL, WTE, TCP, Ping, WLM. Istnieje również możliwość napisania własnego doradcy.

Komponent Interactive Session Support

Interactive Session Support (ISS) jest komponentem współpracującym z Domain Name Server (DNS) w jednym z trzech możliwych trybów pracy: DNS–Update, DNS–Replace lub DNS–Ignore. Komponent ten może być również wykorzystany do zbierania informacji na temat obciążenia serwerów, którą następnie przekazuje do Dispatcher'a. W trybie DNS–Update ISS uaktualnia serwer nazw DNS. W tym trybie działa on w połączeniu z serwerem nazw w celu mapowania nazw DNS na adresy IP serwerów najlepiej nadających się do obsługi danego zadania. W trybie DNS–Replace ISS spełnia dla ograniczonej podgrupy sieci funkcję serwera nazw, nie zawiera jednak wszystkich funkcji standardowego DNS. W trybie DNS–Ignore, ISS działa w połączeniu z Dispatcher'em w celu lepszego równoważenia obciążeń.

Obciążenie na poszczególnych węzłach może być określane poprzez pomiar wykorzystania różnych zasobów, takich jak ilość wolnej pamięci, użycie procesora czy licznik procesów. ISS posługuje się jedną z trzech strategii alokacji zleceń: RoundRobin, Best, i Statistical RoundRobin.

Przy użyciu algorytmu RoundRobin, ISS równoważy obciążenia serwerów w sposób klasyczny dla tej strategii. Z danej grupy serwerów ISS wybiera na określony przedział



Rysunek 5.3: Konfiguracja logiczna modułu Interactive Session Support

czasu jeden serwer, który będzie obciążany w tym przedziale czasu. ISS nie zwraca wówczas uwagi na poziom obciążenia serwera, ale też nie zaleci serwera, który niedomaga. Takie podejście może funkcjonować dobrze, pod warunkiem że obciążenie wywoływane przez klientów jest równomierne. Zaletą tej metody jest brak obciążenia serwerów poprzez procesy pomiarowe. Natomiast jej dużą wadą jest brak elastyczności. Metoda ta może powodować, że np. serwer, który już jest bardzo obciążony, może być dalej preferowany przez ISS, gdyż nie skończył się jeszcze przydzielony mu czas. Stosując strategię Best, w czasie trwania określonego interwału, ISS trasuje żądania do serwera, który na początku tego interwału miał najniższy poziom obciążenia. Tak jak w przypadku poprzedniej metody, korzystający z niej ISS nie zaleci serwera, który przestał funkcjonować. Ta metoda selekcji sprawdza się bardzo dobrze dla każdego czasu trwania połączeń, pod warunkiem, że częstość nowych połączeń będzie relatywnie niska w stosunku do ustawionego interwału czasu przydziału dla pojedynczego serwera. Ta strategia jest przyjmowana domyślnie przez system. W strategii Statistical RoundRobin, ISS równoważy obciążenia na podstawie statystyk obciążenia, które generuje dla wszystkich serwerów. Wykorzystuje te statystyki do budowy profilu najbardziej i najmniej obciążonych serwerów, a następnie w ciągu trwania interwału „heartbeat” rozdziela żądania pomiędzy serwery, proporcjonalnie do ich obciążenia. Również przy użyciu tej metody ISS nie zaleci niedomagającego serwera. Metoda ta sprawdza się wszędzie tam, gdzie występuje duża częstość krótkotrwałych połączeń.

Komponent Content Base Routing

Komponent CBR może być skonfigurowany z WTE (Web Traffic Express) dla serwerów HTTP, lub jako CBR proxy (bez współpracy z WTE) dla serwerów IMAP i POP3. CBR współpracuje wespół z Web Traffic Express, w ten sposób, że klient wysyła żądanie do WTE, który jest skonfigurowany by móc korzystać z CBR; CBR musi być zainstalowany na tej samej maszynie co serwer WTE. WTE wypytuje komponent CBR który serwer ma obsłużyć żądanie. Gdy CBR otrzyma żądanie próbuje je dopasować do ustawionych reguł. Jeśli pasują, CBR wybiera serwer, z grupy skonfigurowanych do odbierania konkretnego typu żądań, jednocześnie równoważąc pomiędzy tą grupą serwerów obciążenie.

CBR jest bardzo podobny w strukturze do pakietu Dispatcher. Trzy kluczowe elementy CBR (Egzekutor, Menadżer i Doradcy) współpracują by zrównoważyć i rozdzielić przychodzące żądania pomiędzy serwery w zbliżony sposób jak w Dispatcherze. Jednakże w przeciwieństwie do modułu Dispatcher – CBR nie oferuje funkcji zwiększonej dostępności. Jednakże można połączyć kilka serwerów CBR i równoważyć ich obciążenie poprzez serwer ND, który to może sprawdzać czy żaden z CBR serwerów nie przestał odpowiadać.

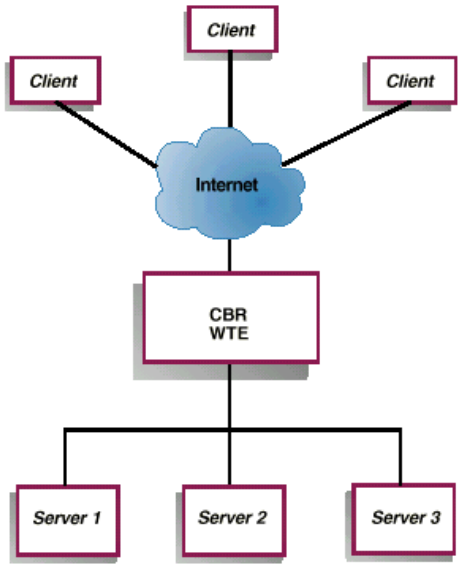
- CBR z WTE (obsługujący ruch HTTP)

  Komponent CBR współpracując z Web Traffic Express działa jako proxy serwer przekierowując rządania klientów do odpowiednich serwerów. Moduł WTE jest proxy serwerem, pozwalającym manipulować pamięcią podręczną w celu szybkiego transferu dokumentów przy niewielkich wymaganiach dotyczących przepustowości sieci. CBR zaś jest w stanie przefiltrować zawartość stron WWW korzystając ze specyficznych ciągów warunków nazywanych rolami. CBR daje możliwość określenia grupy serwerów, które powinny przejąć żądanie opierając się na wyrażeniach regularnych pasujących do zawartości żądania. Ponieważ CBR pozwala na „przywiązanie" poszczególnych serwerów do każdego typu żądania następuje zrównoważenie obciążenia serwerów. CBR potrafi także wykrywać kiedy serwer przestaje odpowiadać wyłączając routing żądań do niego. Zastosowane w module CBR algorytmy równoważenia obciążenia są identyczny z tymi zastosowanymi w komponencie Dispatcher.

  Sposób działania CBR: gdy klienckie żądanie jest przesyłane do WTE proxy następuje tu korelowanie zawartości żądania z regułami zdefiniowanymi w module CBR. Jeśli zbiór reguł pasuje do zawartości żądania jeden z serwerów „związanych" z obsługą tego typu żądań zostaje wybrany do przyjęcia żądania. Wtedy WTE jako proxy serwer przekierowuje do wybranego serwera żądanie. Jak widać moduł WTE musi być uruchomiony zanim CBR zostanie skonfigurowany, ponieważ działa on jako podproces WTE. Oczywiście oba moduły muszą znajdować się na tej samej maszynie.

  Oznacza to podział farmy serwerów na części realizujące odpowiedni typ żądań. Taki podział jest przezroczysty dla klienta. Można np. podzielić witrynę na dwie

części – kilka serwerów realizujących tylko żądania CGI, a pozostałe resztę ruchu HTTP. Pozwala to na realizację intensywnie przetwarzanych skryptów CGI w sposób nie kolidujący z pracą reszty serwerów WWW obsługujących normalny ruch HTTP – co oznacza dla klienta lepszy ogólny czas odpowiedzi. W ten sposób komputery o większej mocy obliczeniowej można przeznaczyć na obsługe normalnego ruchu bez kosztownego upgradu wszystkich komputerów.



Rysunek 5.4: Konfiguracja logiczna modułu Content Base Routing (wraz z WTE)

Inną możliwością podziału serwera WWW jest bezpośrednie przekierowanie klientów, którzy chcą dostać się do stron wymagających rejestracji, do jednego typu serwerów, a resztę do pozostałych. Wtedy obsługą stron wymagających rejestracji zajmują się np. komputery o większej mocy obliczeniowej i klienci, którzy są już zarejestrowani mają lepszy czas odpowiedzi niż inni.

Wybierając role dokonujące równoważenia obciążenia należy sprawdzić czy wszystkie żądania do klastra WWW posiadają role, na podstawie ktorych można wybrać serwer. Jeśli tak nie jest, tzn. jeśli przychodzące żądanie nie pasuje do żadnej roli, klient otrzyma komunikat o błędzie pochodzący z WTE. Najprostszym rozwiązaniem jest stworzenie roli zawsze prawdziwej z bardzo wysokim priorytetem i „przywiązanie” jej do osobnej grupy serwerów.

- CBR proxy (obsługujący ruch IMAP i POP3)

  CBR bez WTE może być proxy wybierającym odpowiedni serwer opierając się na ID użytkownika i haśle. Nie wspiera wtedy opartego na rolach równoważenia obciążeń.

5.2.4 Content Base Routing – konfiguracja

CBR może zostać skonfigurowany jako proxy dla usług opartych o protokoły POP3 lub IMAP – działa wtedy bez współpracy z modułem WTE, lub tez jako narzędzie równoważące obciążenie ruchu HTTP – systemem proxy jest wtedy WTE [**?**].

Moduł ten można skonfigurować za pomocą linii komend lub z pomocą graficznego interfejsu użytkownika (*GUI*). Moduł Content Base Routing jest bardzo podobny w architekturze do Dispatchera. Składa się on z trzech funkcjonalnych części:

- Egzekutor – zajmuje się równoważeniem obciążenia zapytań klienckich. Jest zawsze uruchomiany jeśli CBR jest w użyciu;
- Menadżer – ustala wagi dla poszczególnych serwerów opierając się na:
  - wewnętrznych licznikach Egzekutora;
  - informacjach zbieranych z serwerów przez doradców;
  - informacjach z programów monitorujących prace systemu, takich jak ISS czy WLM.

  Korzystanie z menadżera jest opcjonalne. Jednakże gdy jest on nieużywany – równoważenie obciążenia jest realizowane za pomocą algorytmu statycznego Ważonego Round–Robin opartego na domyślnych wagach, a doradcy nie będą wtedy dostępni.
- Doradcy – wypytują serwery oraz analizują dane o ich stanie, by następnie z tak spreparowanych danych Menadżer ustalił odpowiednie wagi. Korzystanie z Doradców jest również opcjonalne, a typowej konfiguracji wręcz może nie być potrzebne, jednakże mimo to poleca się z nich korzystać.

Wszystkie trzy części (Egzekutor, Menadżer oraz Doradcy) współpracują ze sobą w celu rozbalansowania i rozpropagowania przychodzących żądań pomiędzy serwery.

Aby skonfigurować wstępnie CBR, można posłużyć się jedna z czterech dostępnych metod:

1. Linii komend;
2. Skryptów konfiguracyjnych;
3. Graficznego interfejsu użytkownika;
4. Konfiguracyjnego wizarda.

Konfiguracja maszyny CBR

Aby moc konfigurować CBR, należy mieć status root–a w systemie. Należy także znać adresy każdego z konfigurowanych klastrow serwerów. Pojedynczy adres IP jest tu używany jako jeden adres dla całego klastra. Zanim jednakże przeprowadzi się konfiguracje i uruchomienie modułu CBR – musi być już skonfigurowany i uruchomiony WTE. Konfiguracje wykonuje się w takiej kolejności (konfiguracja dotyczy tylko ruchu HTTP):

1. Uruchomienie WTE
2. Po uruchomieniu WTE, należy zdefiniować klaster WWW, oraz ustawić specyficzne dla tego klastra opcje. Do zdefiniowania klastra oraz opcji służy komenda:

   cbrcontrol cluster set *cluster opcja wartość*

3. Następnym krokiem jest zdefiniowanie portów ustalenie ich opcji. Wykonuje się to komendą:

   cbrcontrol port set *cluster:port opcja wartość*

4. Definiowanie poszczególnych komputerów należących do serwera:

   cbrcontrol server add *cluster:port:server*

   gdzie *server* jest nazwą symboliczną pojedynczego komputera w klastrze, lub jego adresem IP;

5. Następnym krokiem jest konfiguracja reguł CBR. Jest to krok kluczowy w konfiguracji CBR przy ruchu po protokole HTTP reguły (role) definiują jak zadanie URL zostanie przesłane do jednego lub większej ilości serwerów. Te specyficzne reguły nazywają się regułami zawartosci[1]. Aby zdefiniować regułę korzysta się z następującej komendy:

   cbrcontrol rule add *cluster:port:rule* type content pattern=*pattern*

   gdzie wartość *pattern* jest wyrażeniem regularnym, które będzie porównywane za każdym razem jak nadejdzie żądanie od klienta.

[1] *and. – content rule*

6. Następnie należy dodać poszczególne serwery obsługujące reguły zawartości. Gdy występuje dopasowanie pomiędzy żądanym URL, a regułą zawartości zostaje wybrany najlepszy z serwerów obsługujących ten typ żądań. Aby wykonać takie dopasowanie – reguła dopasowania–serwer obsługujący wykonuje się polecenie:

   cbrcontrol rule useserver *cluster:port:rule server*

7. Uruchomienie Menadżera (opcjonalne):

   cbrcontrol manager start

8. Uruchomienie Doradców (opcjonalne):

   cbrcontrol advisor start http *port*

9. Ostatnim krokiem jest skonfigurowanie Doradców, aby ich informacje brały udział w decyzjach równoważenia obciążenia.

Konfiguracja LB opartego na regułach

Wykorzystując do równoważenia obciążenia WTE wraz z modułem CBR – można rozdzielać ruch sieciowy wykorzystując następujące typy reguł:

- Adres IP klienta; jest to sytuacja, gdy wymaga się alokacji zasobów w zależności od tego skąd przychodzi zadanie. Można założyć, ze z pewnych adresów (od klientów) wymagamy by zadania nie były realizowane, zatem przygotowuje się regule i nie dodaje do żadnego serwera – wtedy określeni klienci nie będą obsługiwani (wystąpi błąd). np.:

  ndcontrol rule add 9.67.131.153:80:ni type ip beginrange 9.0.0.0 endrange 9.255.255.255

  taka reguła oznacza, ze klienci z domeny IBM–a nie osiągną serwera CBR;

- Godzina połączenia; taką regułę wykorzystuje się głownie przy określonych planach obsługi obciążeń. W przypadku gdy witryna produkcyjna otrzymuje pewna grupę żądań o pewne dokumenty w określonym (i stałym) czasie każdego dnia – można tylko dla tych żądań wyznaczyć osobne serwery; może to być także przydatne w przypadku gdy w godzinach nocnych (najmniejszy ruch) – niektóre maszyny z powodu wykonywania backupu powinny być wyłączone z realizacji żądań;

- Ilość połączeń na sekundę, na port; (działa tylko podczas uruchomionego Menadżera) można wykorzystywać np. w przypadku: *if połączeń na sekundę na porcie 80 > 100 then użyj te dwa serwery*, *if połączeń na sekundę na porcie 80 > 2000 użyj te 8 serwerów*;

- Całkowita ilość aktywnych połączeń na porcie; (wymóg jak wyżej – Menadżer musi być uruchomiony); Jest to reguła potrzebna w przypadku gdy wiadomo kiedy serwer będzie np. przeładowany (przy ilu połączeniach) Jeśli np. wiadomo, ze serwer nie jest w stanie przyjąć i zrealizować naraz więcej niz. 250 jednoczesnych połączeń tworzy się regułę:

  ndcontrol rule add 130.40.52.153:80:pool2 type active beginrange 250 endrange 500

  Wtedy do tak stworzonej reguły do pojedynczego serwera można dodać następne maszyny, które w przypadku większej ilości połączeń zostaną włączone w podejmowanie żądań;

- reguła zawsze prawdziwe; Reguła ta jest zawsze spełniona – chyba ze serwery z którymi jest związana nie pracują; przydają się w przypadku gdy nie chcemy aby jakikolwiek klient dostał zwrot w postaci błędu zadania dokumentu (od WTE);

- Zawartość zadania; (jest to reguła dostępna tylko z poziomy modułu CBR); reguła wykorzystywana w przypadku dystrybucji żądań w zależności od zawartości zadania; np. gdy potrzeba aby jedna grupa serwerów obsługiwała zadania *cgi–bin*, inna grupa obsługiwała media strumieniowe (np. real media), zaś trzecia wszystkie pozostałe wtedy wzorcem pierwszej z reguł będzie ścieżka do katalogu cgi–bin, drugiej do plików strumieniowych, a trzeciej reguła zawsze prawdziwe; następnie należy tylko przyporządkować reguły do odpowiadających im grup serwerów;

  Typy reguł dla CBR

  składnia wzorców reguł wygląda następująco: w regule (wzorcu) nie może być żadnych przerw (spacji) ani znaków specjalnych:

  ***** – odpowiada 0 do x wystąpienia dowolnych znaków;

)(– używane do grupowania logicznego;

  & – logiczne AND;

  | – logiczne OR;

  ! – logiczne NOT;

  Zarezerwowane słowa (zawsze następuje do nich przyporządkowanie w postaci znaku równości):

  client – adres IP klienta;

url – URL w zadaniu;
path – sekcja ścieżka URL–a;
protocol – sekcja protokółu URL–a;
refer – *quality of service*
Poniżej znajdują się przykłady:

url=http:///*.gif*

*client=9.32.**

!(path=.jpeg)*

(path=index/.gif & protocol=httpd) | (client=9.1.2.3)*

Wszystkie reguły posiadają nazwę, typ, priorytet, początkowy zakres działania, końcowy zakres działania oraz grupę obsługujących serwerów. Dodatkowo, reguła zawartości w komponencie CBR posiada związane ze sobą wyrażenie regularne. Reguły są wykorzystywane w kolejności ich priorytetów. Reguły z niskim priorytetem są realizowane wcześniej. Innymi słowy, reguła z priorytetem 1 jest wykorzystywana przed reguła o priorytecie 2. Pierwsza poprawnie dopasowana reguła zostaje wykorzystana (reszta już nie jest próbkowana).

Aby reguła została spełniona musza być spełnione naraz dwa warunki:

1. Predykat reguły musi być prawdziwy. Co oznacza, ze wartość porównana musi znajdować się pomiędzy wartościami początkowa i końcowa, lub zawartość zadania musi pasować do wyrażenia regularnego wyspecyfikowanego w wzorcu[2]. Dla reguł o typie „zawsze prawdziwy" reguła jest zawsze spełniona (bez warunków);

2. Jeśli istnieją serwery związane do poszczególnych reguł, co najmniej jeden z nich musi być dostępny do odebranie zadania.

W przypadku, gdy nie ma serwerów dla których konkretna reguła nie mogłaby być spełniona – zadanie zostaje porzucone, a CBR zwróci do serwera proxy (WTE) komunikat błędu. Jeśli zaś żadna z reguł nie może zostać spełniona – jak wyżej – CBR zwróci do WTE komunikat błędu.

5.2.5 Narzędzie do testów – Astra Load Runner

Narzędziem testowym w wykonanym projekcie było oprogramowanie firmy Mercury Interactive[3] – Astra Load Runner. Jego wybór był spowodowany szerokim wachlarzem moż-

[2] ang. *pattern*
[3] http://www.merc-int.com

liwości testowych, obsługiwanych protokołów oraz możliwości dokonywania analiz i wizualizacji rezultatów. Poniżej znajduje się krótka charakterystyka tego progrmau:

LoadRunner jest narzędziem do testów wydajnościowych, dzięki którym można sprawdzić wydajność i skalowalność systemu webowego. Jest on w stanie generować dziesiątki, setki czy nawet tysiące jednoczesnych użytkowników, umożliwiając w ten sposób wykrycie wszystkich słabych, bądź niewydajnych miejsc w naszym systemie. W czasie trwania testu dostępne są różnorakie monitory, wyświetlające cały czas wszystkie interesujące parametry systemu funkcjonującego pod obciążeniem. Dzięki tym wbudowanym monitorom można np.: na bieżąco śledzić czasy odpowiedzi wszystkich serwerów, czasy wykonywania się danych transakcji czy szybkość transferu danych w sieci z podziałem na segmenty.

Wszyscy użytkownicy generowani przez LoadRunnera kontrolowani są z jednego centralnego modułu zwanego kontrolerem. Testując system pod obciążeniem można symulować różne numery IP klientów, różne przeglądarki internetowe oraz różne szybkości łączy internetowych, cały czas jednocześnie monitorując zachowanie naszego obciążanego systemu. Dane Techniczne Narzędzia:

Wspierane protokoły typu klient/serwer

- Oracle UPI
- Oracle OCI
- ODBC
- MS-SQL Server
- Sybase ctlib
- Sybase dblib
- Informix I-Net
- DB2 CLI
- Tuxedo (including compression mode)
- RTE
- CORBA
- COM/DCOM
- WinSocket

Wspierane protokoły ERP

- SAP R/3
- PeopleSoft (2-tier i Tuxedo-based)
- Oracle Applications

- Siebel
- Baan

Wspierane protokoły internetowe

- Streaming (Real Audio and Real Video)
- MS Media
- iMode
- VoiceXML
- LDAP
- WAP–HTTP
- WAP–Gateway
- HTTP
- HTTPS (SSL)
- Digital Certificates
- RMI
- FTP
- POP3
- Winsock
- SMTP

Wspierane systemy Legacy

- 3270 Terminals (Mainframe)
- 5250 Terminals (AS/400)
- VT Terminal (DEC)
- X Window Applications

Tworzenie testów wydajnościowych:

- Możliwość przełączania rodzaju nagrywanego protokołu w trakcie rejestracji jednego skryptu;
- Narzędzie automatycznie generuje skrypty testów obciążeniowych, oparte na operacjach biznesowych wykonywanych na testowanej aplikacji;
- Narzędzia do automatycznej parametryzacji skryptów pozwalają na szybką i wydajną obróbkę stworzonego testu;

- Przypisywanie różnych numerów IP do poszczególnych symulowanych użytkowników daje duże możliwości w tworzeniu scenariuszy testowych;
- Automatyczna kontrola zawartości danych w czasie testów wydajnościowych, stanowiąca jednocześnie kontrolę funkcjonalną;
- Automatyczne korelowanie dynamicznie zmieniających się danych przy nagrywaniu i odtwarzaniu skryptów;
- Korelacja zapytań Oraclowych, polegająca na możliwości przechwytywania danych pobieranych z bazy i wykorzystywaniu ich w dalszej części skryptu;
- Emulacja połączenia modemowego o różnej przepustowości;
- Kreator scenariuszy testowych, dający możliwość szybkiego i efektywnego tworzenia całych scenariuszy testowych.

Kontrolowanie testów wydajnościowych

- Automatyczna kontrola i synchronizacja wszystkich symulowanych użytkowników z jednego centralnego punku kontroli;
- Monitorowanie w czasie rzeczywistym wszystkich parametrów systemu;
- Łatwy w użyciu graficzny interfejs, umożliwiający tworzenie, uruchamianie, monitorowanie i analizowanie wyników testu;
- Jeden wspólny punk kontroli dla całego testu z możliwością uruchamiania użytkowników na systemach Windows i Unix.

Pomiary i kontrola wydajności

- Możliwość pomiaru i kontroli czasu wykonywania się mierzonych transformacji w zadanych kryteriach lub od „zapytania – do odpowiedzi" tzn. łącznie z czasem przetwarzania klienta, serwera i sieci komputerowej;
- Możliwość wygenerowania dedykowanego obciążenia serwera, powielając zarejestrowaną transmisję po konkretnym protokole;
- Pomiar parametrów systemu operacyjnego działającego na obciążanym serwerze, dający możliwość znalezienia słabych punktów konfiguracji i zasobach systemu operacyjnego;
- Pomiar parametrów sprzętowych maszyny będącej obciążanym serwerem, dający możliwość znalezienia słabych i niewydolnych punków w konfiguracji sprzętowej testowanej maszyny;
- Możliwość pomiaru czasów wykonywania się transakcji zdefiniowanych przez testera.

Analiza i kontrola wyników, dostępne monitory:

- Server Resource Monitor (NT, Unix, Linux) – pozwala na kontrolowanie zasobów sprzętowych maszyn będących testowanymi serwerami;
- Network Delay Monitor – pozwala na kontrolowanie czasów odpowiedzi i realizowania się monitorowanych transakcji we wszystkich segmentach sieci obciążanego systemu;
- SNMP Monitor – pozwala na kontrolowanie przepływu danych pomiędzy poszczególnymi elementami aktywnymi sieci jak: Routery, Huby, Bridge;
- WebSphere, ColdFusion, SilverStream, BroadVision, ATG Dynamo, GemStone/J, Ariba Buyer, WebLogic, MS ASP, iPlanet – monitory serwerów aplikacji internetowych, posiadające specjalne wsparcie dla każdego z nich;
- EJB, TUXEDO – kontroluje i monitoruje wszystkie parametry systemu opartego na aplikacjach TUXEDO i EJB;
- MS IIS, Netscape, Apache – monitory serwerów webowych;
- RealServer, Windows Media Server – monitory do kontrolowania technologii strumienowych;
- COM/CORBA – monitory do kontrolowania i badania technologii rozproszonych;
- ORACLE, SQLServer – monitory do śledzenia i badania wydajności serwerów baz danych, zawierające specjalne wsparcie i integracje z tym technologiami.

Wirtualni użytkownicy są odtwarzani na systemach:

- Windows NT
- Sun Solaris
- HP–UX
- IBM AIX
- Linux

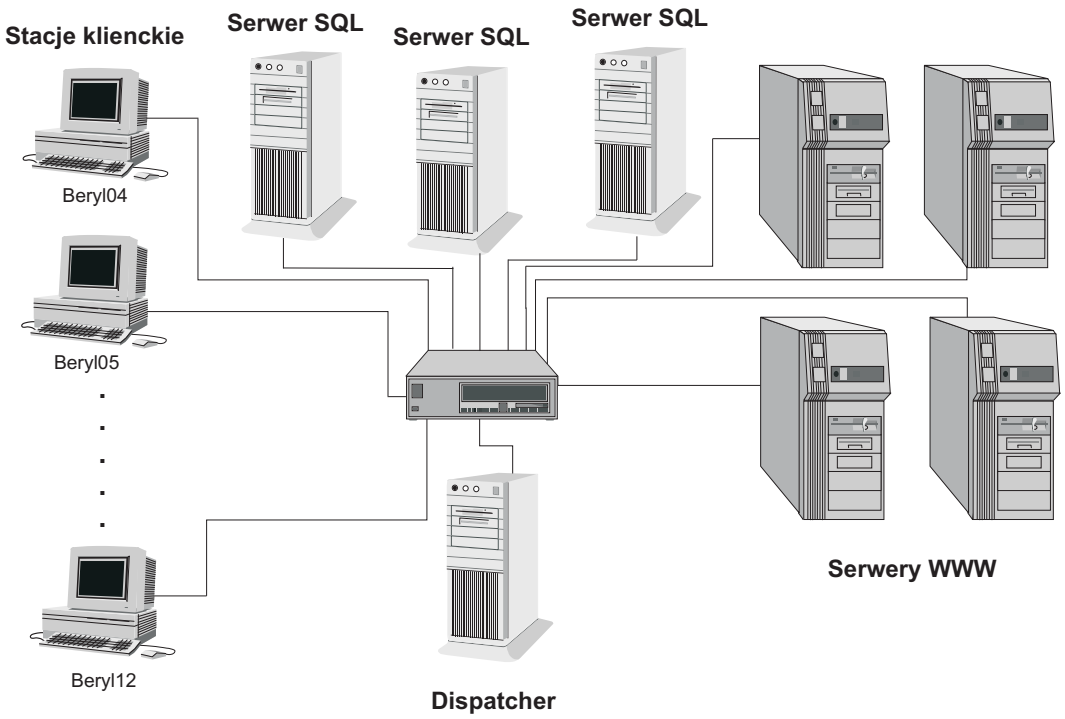
5.3 Projekt systemu do zarządzania wielokomputerowymi serwerami WWW w oparciu o IBM SecureWay Network Dispatcher

Całość zadania polegała na stworzeniu jednej architektury systemu WWW którego witryna zawierałaby około 60% zawartości w postaci stron dynamicznych (wypełnianych danymi pochodzącymi ze znajdującego się w układzie serwera baz danych) generowanych

on line. Jednakże rozdysponowanie obciążenia pomiędzy poszczególnymi elementami serwisu WWW było realizowane na trzy sposoby: statycznie, dynamicznie oraz w zależności od typu nadchodzących do serwera żądań. Taka architektura miała dać odpowiedź na pytanie – jak zaprojektować sytem WWW obsługujący strony generowane dynamicznie w możliwie najbardziej efektywny sposób.

5.3.1 Architektura

W eksperymencie wykorzystano lokalną sieć komputerową złożoną z jednego segmentu sieci FastEthernet. W jego skład wchodziły komputery, na których zainstalowane było oprogramowanie generujące obciążenie dla klastera serwerów WWW (razem dziewięć sztuk) pracujących pod kontrolą systemu Windows NT 4.0 Server (Service Pack 6a).



Rysunek 5.5: Architektura segmentu sieci testowej.

Każdy komputer wyposażony był w procesor Intel Celeron 300 MHz, 96 MB pamięci SDRAM, dysk twardy o pojemności 3 GB oraz kartę sieciową 3COM EtherLink XL (10/100 Mbps). Płyta główna każdego komputera zbudowana była w oparciu o chipset Intel 430 LX. Komputery te posiadały adresy IP z zakresu 156.17.130.69 – 156.17.130.79 i nazwy odpowiednio Beryl04 – Beryl12. Na komputerach tych zainstalowane było oprogramowanie Astra LoadRunner firmy Mercury Interactive. Komputery o identycznej architekturze stanowiły serwery baz danych: beryl01 (IP = .66), beryl02 (IP = .67), beryl03

(IP = .68). Zainstalowano na nich MS SQL server w wersji 7.0 (wersja 120 dniowa) wraz z bazą testową.

Poza wyżej wymienionymi komputerami PC podłączony były badany klaster serwerów WWW. Sprzętową platformą serwerów były komputery IBM RS 6000 43P Model 260 pracujące pod kontrolą systemu operacyjnego AIX 4.3.3. Każdy z tych komputerów posiadał 256 MB pamięci RAM, dysk twardy UW SCSI o pojemności 4.3 GB oraz zintegrowaną kartę sieciową IBM 10/100 Mbps. „Sercem" tej serii komputerów RS 6000 jest procesor RISC PowerPC Power3 taktowany częstotliwością 200 Mhz. Trzy z wykorzystywanych maszyn posiadały jeden taki procesor a dwa wyposażone były w dwa procesory połączone w architekturze SMP. Na każdym z badanych serwerów zainstalowane było oprogramowanie serwera WWW Apache for AIX. Poszczególne komputery RS 6000 posiadały adresy IP z zakresu 156.17.130.45 – 156.17.130.49 i nazwy odpowiednio akwamaryn (RS/6000 dwuprocesorowy), szafir (dwuprocesorowy), opal01 – opal03 (jednoprocesorowe). Jedna maszyna dwuprocesorowa (szafir) posłużyła do rozpropagowania obciążenia (dispatcher+CBR i WTE), druga posłużyła do obsługi żądań o statyczne pliki HTML, zaś pozostałe maszyny obsługiwały dynamicznie generowane (poprzez skrypty PHP) strony, w ten sposób, że każdy z opali pobierał dane z bazy danych poszczególnych beryli (opal01–beryl01, opal02–beryl02 i opal03–beryl03).

Wszystkie komputery były połączone ze 100Mbps switchem OmniS/R-5 firmy Alcatel. Również wyjście na Swiat było realizowane z tą szybkością.

Konstrukcja zarządzanego, wielokomputerowego systemu WWW była bardzo złożona. Składała się z następujących elementów:

1. Instalacja i konfiguracja switcha OmniSwitch/Router firmy Alcatel;
2. Instalacja i konfiguracja dispatchera z modułem CBR. W skład tej części weszły:
   - instalacja systemu AIX w wersji 4.3.3 oraz jego update do wersji 4.3.3.0.8 (niezbędne pliki zostały ściągnięte z odpowiedniej strony firmy IBM;
   - instalacja wirtualnej maszyny (VM) Javy w wersji 1.3.0;
   - instalacja oprogramowania IBM WebSphere Edge Server 1.0.3 wraz z 60–dniową licencją;
   - skonfigurowanie WTE jako *reverse proxy* o adresie URL: www1.ists.pwr.wroc.pl, który przekierowywał adresy URL do serwerów: opal01, opal02, opal03 i akwamaryn, w ten sposób, aby żądania o statyczne pliki HTML zostały skierowane do serwera akwamaryn, zaś żądania o dynamicznie generowane strony z rozszerzeniami: .PHP, .PHP3 i PHP4 zostały rozpropagowane na pozostałe trzy maszyny;
3. Instalacja i konfiguracja oprogramowania dodatkowego na poszczególne serwery WWW i dispatchera tzn. HTTP Server Apache w wersji 1.3.19, kompilator C i C++ w postaci gcc w wersji 2.95.3 oraz języka skryptowego PHP w wersji 4.0.6;

4. Instalacja i konfiguracja Windows NT 4.0 Server wraz z Service Pack 6.0 na dwunastu laboratoryjnych komputerach PC;
5. Instalacja i konfiguracja Microsoft SQL Serwera w wersji 7.0 na serwerach baz danych oraz instalacja bazy testowej;
6. Instalacja oprogramowania do testów na pozostałych komputerach; przygotowanie testu;
7. przygotowanie witryny WWW do testów;

5.3.2 Algorytmy i metody

Poniżej znajdują się opisane trzy powstałe modele. W każdym z nich program komputer (RS/6000) wraz z zainstalowanym programem IBM SecureWay Network Dispatcher stanowi układ rozdysponujący żądaniami.

Information less – moduł dispatcher + Round Robin

Sam Dispatcher nie ma możliwości pracy w statycznym algorytmie Round Robin, jednakże (jak napisano powyżej, w charakterystyce oprogramowania) można posługiwać się modułem Dispatcher z wyłączonym modułem Zarządcy oraz z ogórnie nadanymi wagami. W tym przypadku, pomimo niehomogenicznego środowiska nadano wszystkim elementom klastra WWW wagi równe jeden. W tej sytuacji komputer z zainstalowanym dispatcherem równomiernie (bez jakiejkolwiek analizy) propagował żądania do poszczególnych komputerów.

W tym przypadku dostęp do bazy danych mógł być realizowany z dowolnego komputera (serwera WWW).

Server info aware – moduł dispatcher + Weighted Round Robin

W porównaniu do poprzedniego przypadku w tym użyto Dispatchera z uruchomionym modułem zarządcy. Wtedy też realizowany był algorytm Ważony Round–Robin. Wagi nadawane były dynamicznie tzn. w zależności od obciążenia poszczególnych komputerów – serwerów WWW.

W tym jak i w powyższym przypadku dostęp do bazy mógł być również realizowany z dolwolnego z serwerów WWW.

Client info aware – moduł CBR + WTE

Był to najbardziej złożony przypadek. Każdy pakiet był analizowany pod względem typu zawartości oraz pod względem jego obecności w keszu komponentu WTE. CBR został tak

skonfigurowany aby rozpoznawać URL z zawartym w nim wyrazem .php (identyfikującym plik HTML z zawartym kodem PHP dostępu do bazy). Pozwoliło to na rozdzdział zapytań na żądania o pliki statyczne (te były rozsyłane go trzech jednoprocesorowych maszyn RS/6000), zaś te żądania o pliki, których część była generowana dynamicznie były przekazywane na dwuprocesorowy serwer, który komunikował sie (tylko on) z serwerem baz danych. Taka konfiguracja pozwoliła rozdysponować zarówno obciążenie jak i typ połączeń.

5.3.3 Metodologia testowania

Celem testowania powstałego systemu do zarządzania wielokomputerowym serwerem WWW jest zbadanie wydolności takiego systemu – tzn. jego wydajności w sytuacji maksymalnego obciążenia, czyli także możliwości odpowiedzi na klienckie żądanie w takiej sytuacji. Zamierzeniem testowania jest odpowiedź na pytanie jak obciążenie jest równoważone pomiędzy poszczególnymi częściami składowymi systemu, w zależności od algorytmu dysypacji obciążenia (Round–Robin, Weighted Round–Robin oraz rozproszenie oparte na CBR), oraz jaki dany algorytm ma wpływ na LB.

Testowanie zaprojektowanego rozproszonego serwera WWW polegało na pomiarze wydajności tak wejściowej jak i wyjściowej całego systemu jak i poszczegolnych jego składowych (poszczególnych serwerów WWW, serwera baz danych i dispatchera).

Celem testowania było zbadanie maksymalnej wydajności systemu. Wydajność systemu można badać na kilka sposobów:

- ilość zapytań w czasie,
- ilość danych przesłanych w czasie.

Na wynik wpływ ma informacja w jaki sposób rozłożone były zapytania oraz czy sesje kończyły się prawidłowo. Założono trzy możliwości odpowiedzi systemu:

- w czasie do 3 s.,
- w czasie 5 s.,
- odpowiedz powyżej 8 s.

Za sesję prawidłowo zakończoną uznno taką, podczas której dwie kolejne odpowiedzi na pytania nie przekroczyły 8 s.

Założono następujące ścieżki poruszania się po serwisie [**?**]:

- *Home* —> *Exit* (Home – strona domowa);
- *Home* —> *First Level* —> *Exit* (First Level – np. informacja o produkcie);

- *Home —> First Level —> Object —> Exit* (Object – np. zakup).

Przy założeniu, że ruch na „typowej" witrynie [**?**, **?**, **?**] wygląda w następujący sposób:

- *Home* – 58 %;
- *First Level* – 31 %;
- *Object* – 11 %;

Na bazie powyższych danych należy zbudować skrypty generujące zapytania.

Analiza wyników

Aby prawidłowo odpowiedzieć na pytanie związane z wydajnością i dostępnością systemu należy podać, przy jakiej ilości użytkowników system jest w stanie prawidłowo odpowiadać czyli przy x użytkowników (otwartych sesji) system odpowiada (pomiary zostają wykonanywane w określonych przedziałach czasowych):

- % w czasie do 3 s.
- % w czasie do 5 s.
- % powyżej 8 s.

Jeżeli podczas sesji dwie następujące po sobie odpowiedzi przekroczyły 8s. – oznacza to, że ich sesje zostały zerwane. System nie może być obciążony w większym stopniu niż aktualny aby wszystkie napływające do niego sesje zostały prawidłowo obsłużone.

Tak zaplanowane testy należy wykonać podczas pracy Dispatcher–a z różnymi algorytmami: Round Robin, Multi Class Round Robin, Weighted Round Robin. Wyniki powyższych testów pozwalają sprawdzić, jak zachowuje się system w przypadku chwilowego dużego obciążenia.

Kolejnym elementem testów byłoby zasymulowanie normalnych warunków pracy systemu na granicy 80 % wydajności. Podczas takiej pracy należałoby wygenerować obciążenie znacznie przewyższające możliwości systemu. Obciążenie to powinno trwać 3s, 5s, więcej niż 8s. Pozwoli to sprawdzić jak system reaguje i jak szybko jest w stanie się ustabilizować doprowadzając do równego rozłożenia obciążenia na poszczególne serwery.

5.4 Wyniki wstępnych eksperymentów

Na skutek poważnych problemów, tak sprzętowych jak i programowych autor pracy nie zdołał zrealizować wszystich zaplanowanych zadań. Poniżej wymieniono kolejne elementy budowy systemu do zarządzania wielokomputerowym serwerem WWW oraz stopień ich realizacji.

1. przygotowano środowisko serwerów front–endowych oraz dispatchera:
   - uruchomiono oraz zainstalowano system operacyjny AIX 4.3.3 na komputerach IBM RS/6000, ktore miały stanowić serwer dispatchera, oraz cztery serwery WWW; na tych również maszynach wykonano niezbędny update do wersji 4.3.3.0.8;
   - zainstalowano menadżer pakietów RPM, kompilator gcc 2.95.3, Apache 1.3.19 oraz język skryptowy PHP 4.0.6 (za jego pomocą należało stworzyć interface pomiędzy serwerem WWW a serwerem baz danych) – jako moduł ładowalny do serwera Apache;
   - skonfigurowano każdy z serwerów WWW wraz z modułem ładowalnym PHP;
   - zainstalowano i skonfigurowano IBM WebSphere Edge Server jako *reverse proxy* wraz z modułem CBR odpowiedzialnym za specyficzne, oparte na regułach przekierowywanie żądań; podczas instalacji napotkano jednakże na dwa poważne problemy, które uniemożliwiały instalację, a tym samym działanie pakietu: moduł dispatcher nie uruchamiał się zgłaszając brak pliku licencji, zaś WTE nie uruchamiał się z powodu niekompatybilności z główną biblioteką systemową (libc.a) (na obie sytuacje nie znaleziono pomocy w żadnej posiadanej literaturze); skorzystano z *supportu* firmy IBM; w pierwszym przypadku – otrzymano poprawny plik licencyjny, zaś w drugim rozwiązaniem okazało się przełączenie urządzeń I/O w tryb asynchroniczny; po wykonaniu obu zadań oraz skonfigurowaniu modułów – dispatcher + WTE działało poprawnie;
2. przygotowano środowisko serwerów back-endowych (baz danych):
   - zainstalowano na trzech komputerach PC system operacyjny Windows NT 4.0 Server wraz z serwerem baz danych Microsoft SQL Server 7.0 (wersja 120–dniowa), następnie skonfigurowano te maszyny;
   - zainstalowano bazę testową na serwerach;
3. system testowy Astra LoadRunner; niestety nie udało się uzyskać tego oprogramowania testowego z powodu braku wersji testowej (komercyjna kosztuje około 13000$); uzyskano jednakże program Astra LoadTest o zbliżonej funkcjonalności (wersja testowa na dzisięciu wirtualnych użytkowników działająca bez ograniczeń 7 dni); został zainstalowany i sprawdzony w kierunku wymagań tej pracy; wykonano za jego pomocą wstępny, przykładowy test, opis i wyniki znajdują się w Dodatku;

Po poprawnym wykonaniu wyżej wymienionych czynności autor pracy stanął przed problemem pobierania danych z serwerów baz danych przez serwery WWW. Pakiet RPM języka PHP zawierał binarny moduł ładowalny do Apache skompilowany bez wsparcia w komunikacji z bazami danych (poza samym modułem ładowalnym istnieją jeszcze moduły

odpowiadające komunikacji z poszczególnymi serwerami baz danych). W związku z tym, że PHP jest językiem, który jest dostępny w źródłach (na licencji GPL) próbowano poprzez kompilację i odpowiednią konfigurację włączyć go do Apache. MS SQL Server jest serwerem baz danych opartym na motorze firmy Sybase, dlatego aby nawiązać komunikację z serwera WWW do serwera bazy danych MS SQL poprzez PHP należało przygotować (przekompilować) odpowiednie moduły (klienty) PHP w formie modułów ładowalnych. Niestety z powodu braku odpowiedniego API nie udało się tego zadania zrealizować (aby móc stworzyć klienty PHP do baz danych – należy kompilować PHP wraz z API odpowiedniego serwera bazy danych).

Próbowano także skorzystać z odpowiednich sterowników ODBC firmy OpenLink[4]. Poza ograniczeniami w korzystaniu z tego oprogramowania (możliwość pracy tylko dwóch jednoczesnych użytkowników podczas sesji) nie udało się przekompilować PHP (występowały błędy podczas kompilacji). Podczas poszukiwania rozwiązania w Internecie, skorzystano również z możliwości Usenetu (listy dyskusyjne).

Powyższego problemu nie udało się również rozwiązać analizując zmianę języka oprogramowania: PERL (kolejny znakomity język skryptowy umożliwiający generowanie stron dynamicznych) – ma zbliżone wymagania jeśli chodzi o klientów baz danych, ASP (nie istnieje na maszyny Unixowe), C, C++ lub Java (duże kłopoty implementacyjne, spora pracochłonność).

W związku z brakiem możliwości scalenia sewerów WWW z serwerami baz danych nie udało się przetestować w pełni działającego systemu do zarządzania wielokomputerowym serwisem WWW.

[4]http://www.openlinksw.com

Dodatek

Test

W związku z awarią sieci teleinformatycznej w Instytucie Sterowania i Techniki Systemów poniższy próbny test przeprowadzono na skonstruowanym wielokomputerowym systemie WWW za pomocą tylko jednej maszyny komunikującej się z systemem za pomocą 10MB huba (firmy IBM).

Na kolejnych stronach znajdują się wynikowe zestawienia i wykresy próbnego testu wykonanego za pomocą oprogramowania Astra LoadTest. Test został wykonany z dziesięcioma wirtualnymi użytkownikami[5] (uruchamianymi w kolejności co 15 sek.) poruszającymi się po znajdującej się na systemie WWW dokumentacji do serwera Apache (razem dziesięć dowolnie wybranych stron). Trwał on 6 min. i 19 sek. Jak widać na rys. 5.6 łącznie przesłano: 19915485 bajtów ze średnią przepustowością na poziomie 52 kb/sek; łącznie 1925 trafień (średnio pięć na sekundę). Również na rys. 5.6 znajduje się tabela transakcji wraz ze średnim, minimalnym i maksymalnym czasem jej realizacji.

Na rys. 5.7 widać wykres przepustowości serwisu WWW w bajtach na sekundę. Poniżej wykresu znajduje się jego charakterystyka wraz z odchyleniem standardowym i medianą.

Kolejny rysunek (rys. 5.8) przedstawia wykres ilości trafień na sekundę (charakterystyka wykresu poniżej).

Następny wykres (rys. 5.9) przedstawia zależność ilości transakcji od pobranej strony.

Na wykresie (rys. 5.10) widać średni czas realizacji transakcji (każdej strony). Rys. 5.11 zawiera zestawienie średnich, najkrótszych i najdłuższych czasów realizacji transakcji, a rys. 5.12 procentowy rozkład ilości transakcji w czasie.

Wykres na rysunku 5.14 obrazuje ilość pobranych stron na sekundę.

[5] *Vusers*

Analysis Summary Period: 06-10-2001 21:20:51 - 06-10-2001 21:27:10

Scenario Name: Scenario1
Results in session: C:\tesciki\kamil_nowy1\LrRes\LrRes.lrr.
Duration: 06 minutes and 19 seconds.

Statistics Summary

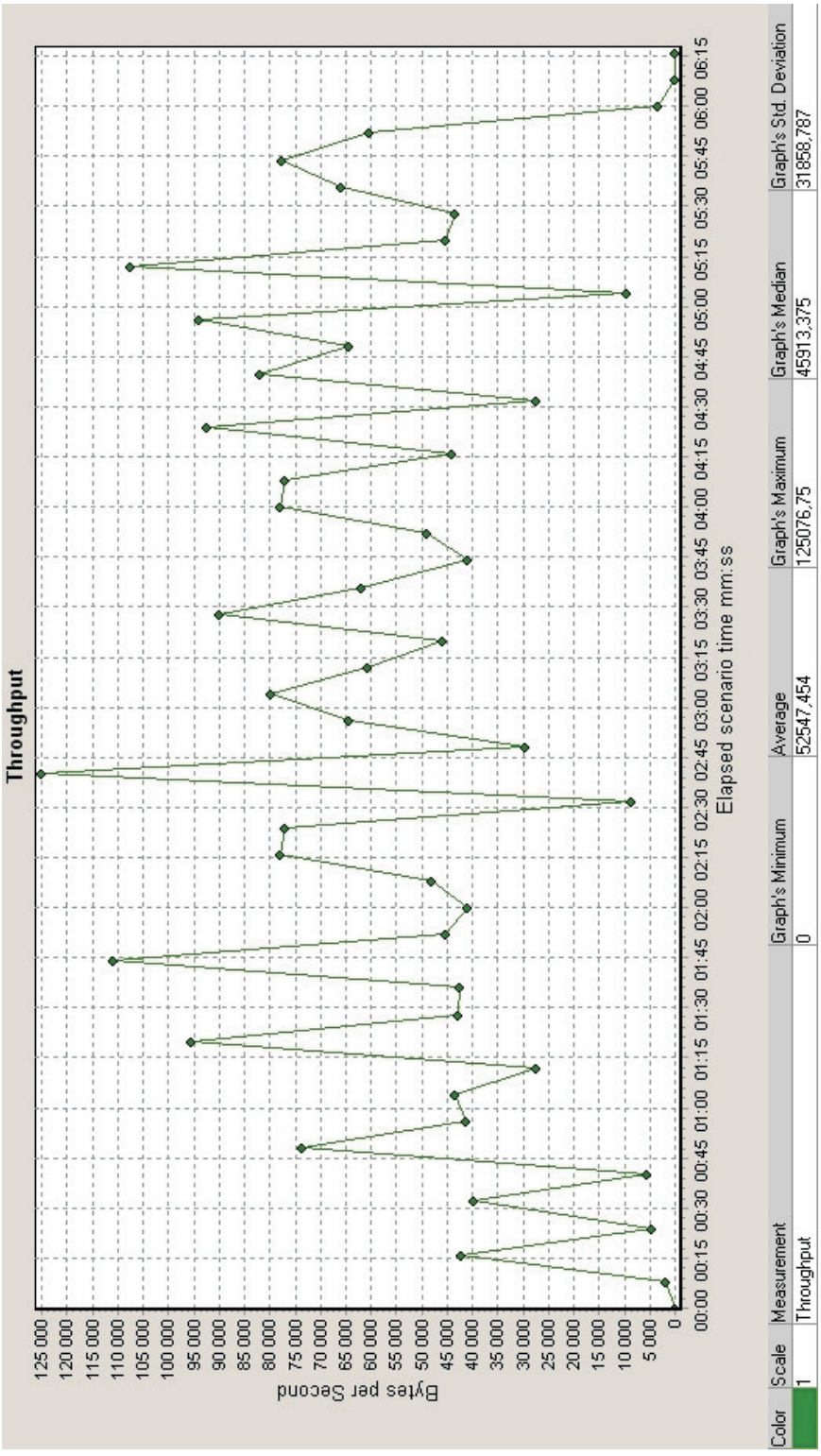
Maximum Running Vusers:	10
Total Throughput (bytes):	19 915 485
Throughput (bytes/second):	Average: 52 547
Total Hits:	1 925
Hits per Second:	Average: 5

Transaction Summary

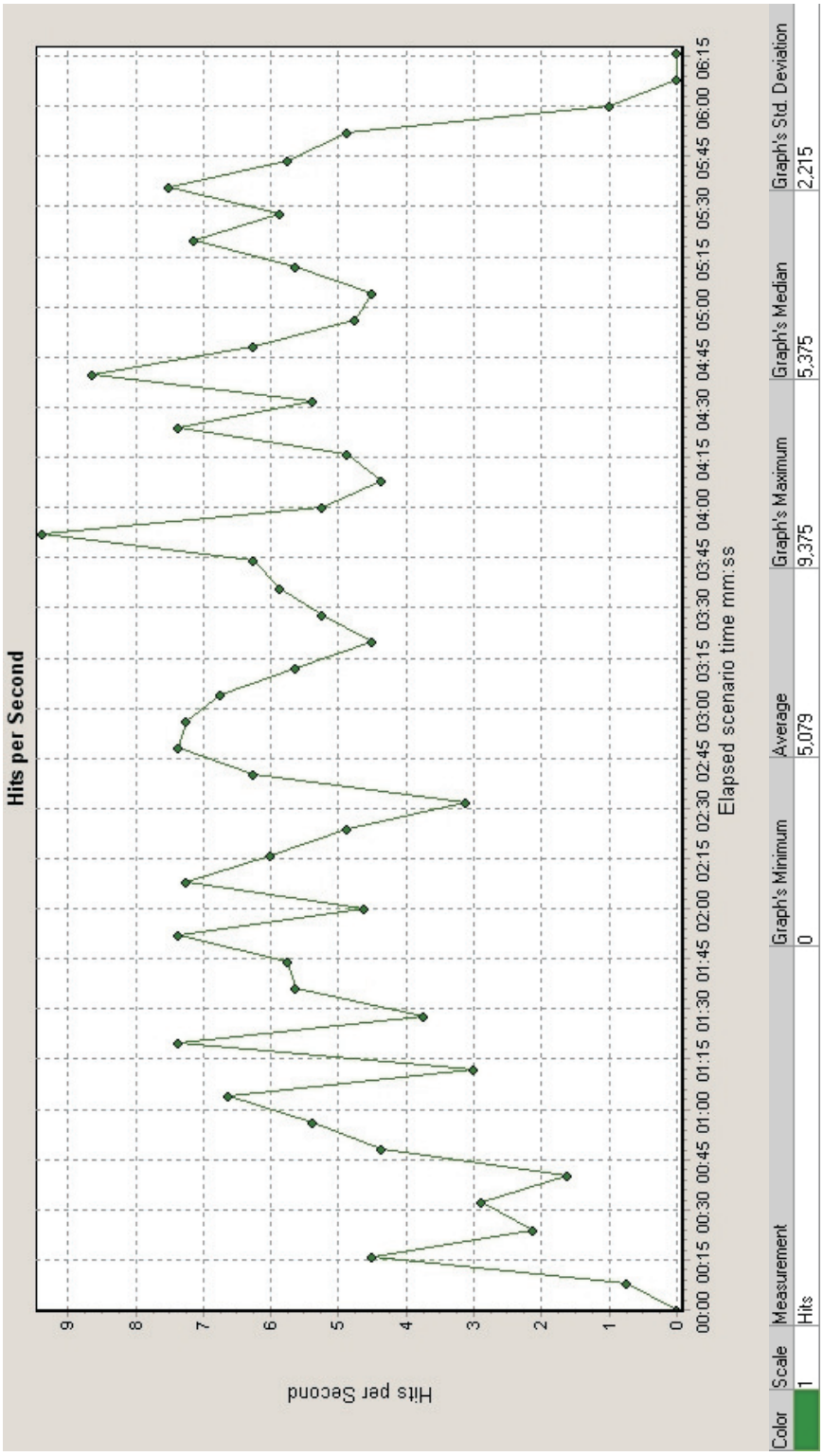
Transactions:	Total passed: 684	Total failed: 0	Total aborted: 0	Response Time Avg.

Transaction Name	Minimum	Average	Maximum	90 Percent	Pass	Fail	Abort
Apache_Core	0,02	0,173	0,621	0,34	64	0	0
Apache_Core_2	0,04	0,219	0,671	0,4	57	0	0
Apache_directives	0,03	0,22	0,851	0,51	65	0	0
Apache_HTTP	0,05	0,229	1,031	0,44	66	0	0
Apache_module	0,02	0,157	0,831	0,3	64	0	0
Definitions	0,18	0,37	1,031	0,68	64	0	0
Definitions_2	0,03	0,261	1,102	0,59	58	0	0
How_Directory,	0,19	0,384	1,182	0,66	58	0	0
http://www1.ists.pwr.wroc.pl/	0,03	0,167	0,891	0,37	56	0	0
StartUp	0,02	0,119	0,651	0,37	66	0	0
Strona_testowa	0,04	0,24	0,971	0,41	66	0	0

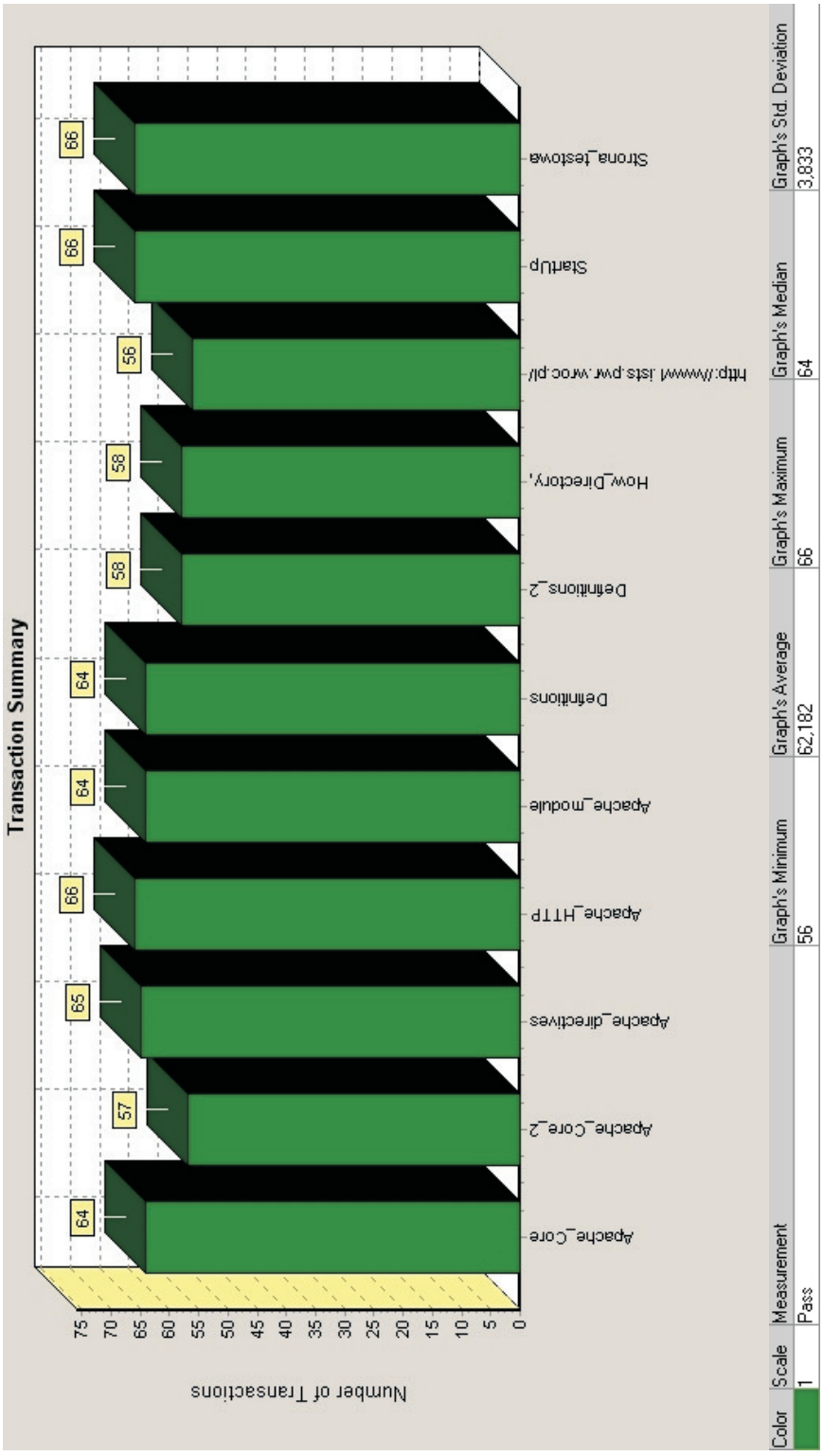
Rysunek 5.6: Zebrane informacje o przebiegu testu



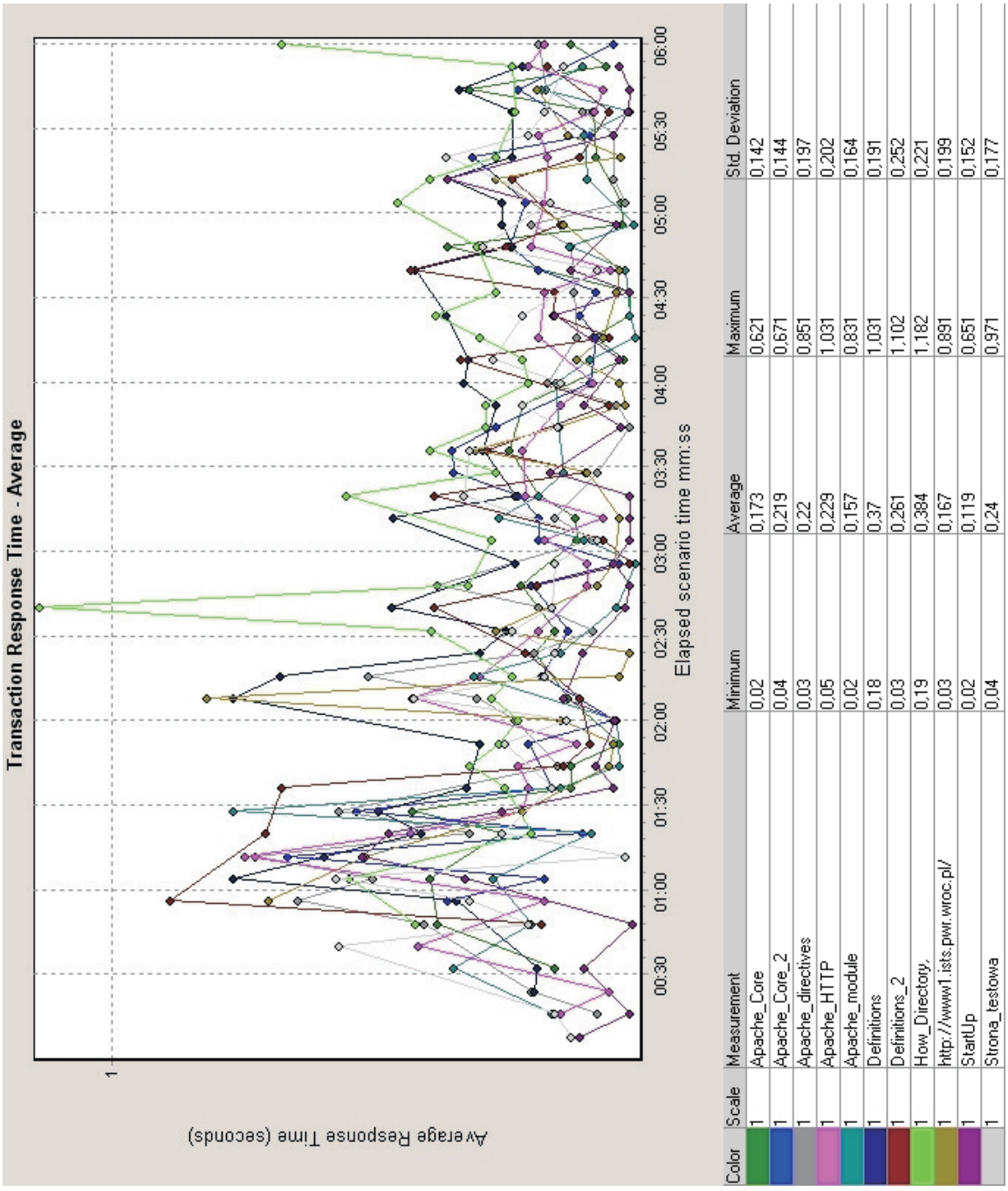
Rysunek 5.7: Wydajność wielokomputerowego systemu WWW



Rysunek 5.8: Ilość realizacji żądań na sekundę

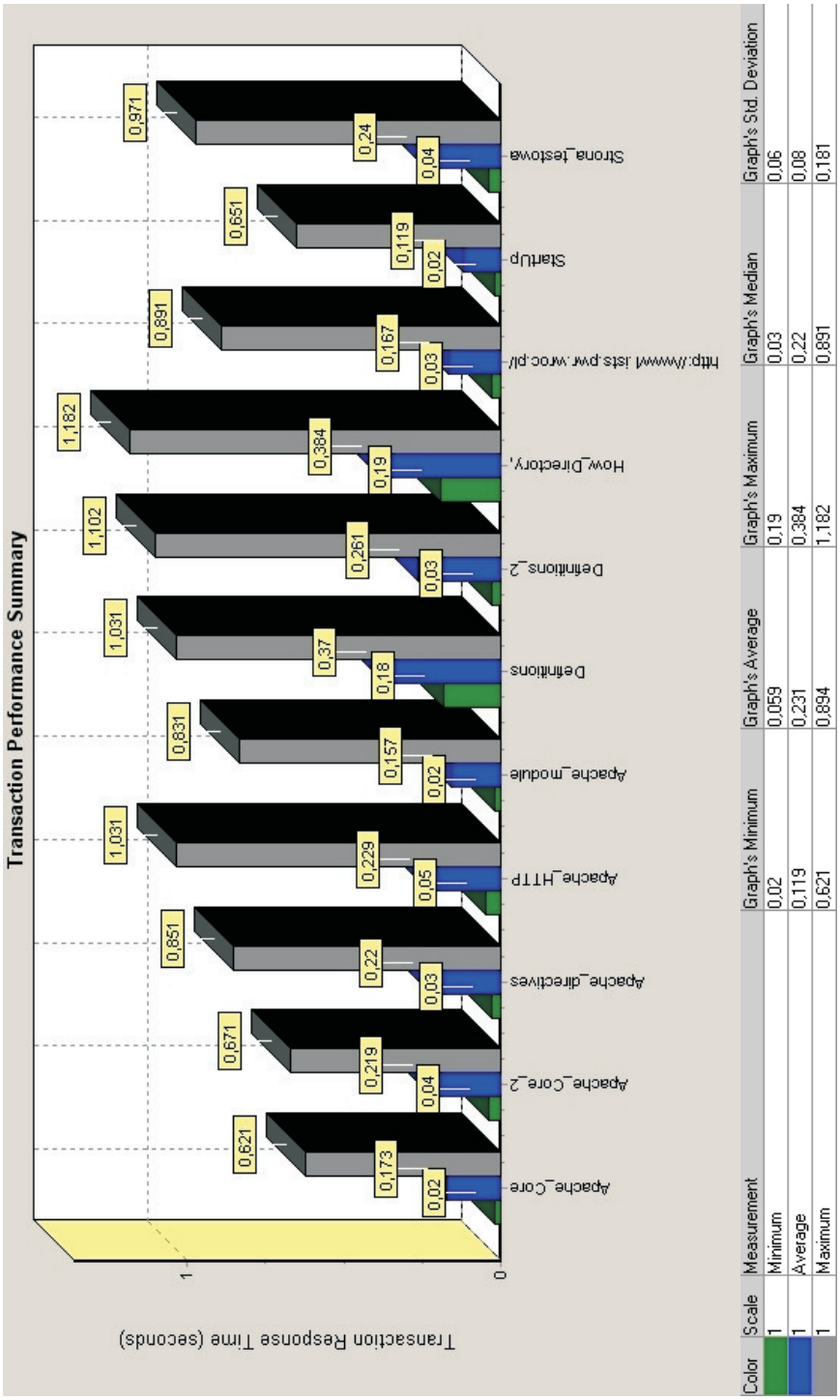


Rysunek 5.9: Ilość transakcji w zależności od pobieranej strony

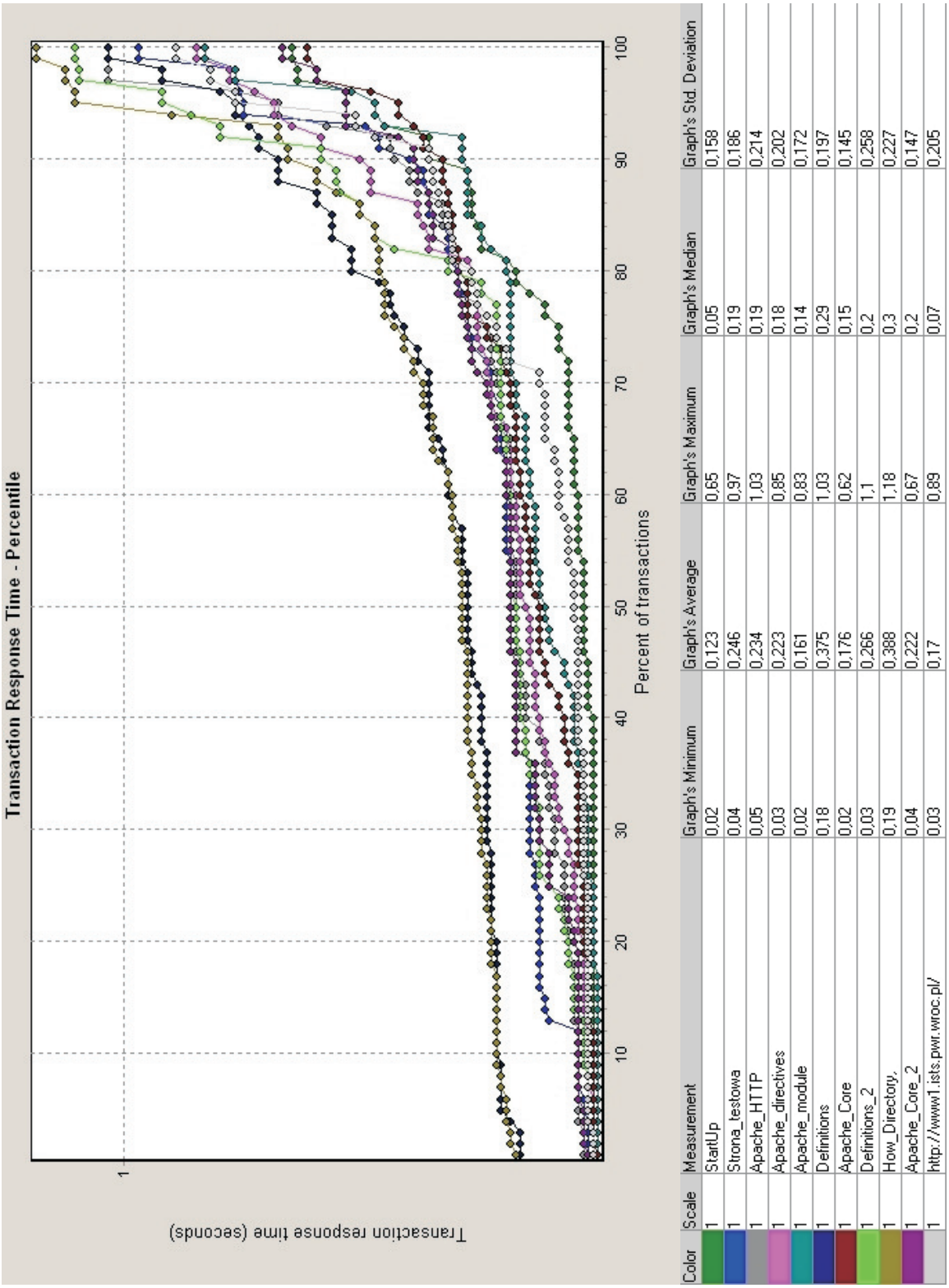


Color	Scale	Measurement	Minimum	Average	Maximum	Std. Deviation
	1	Apache_Core	0,02	0,173	0,621	0,142
	1	Apache_Core_2	0,04	0,219	0,671	0,144
	1	Apache_directives	0,03	0,22	0,851	0,197
	1	Apache_HTTP	0,05	0,229	1,031	0,202
	1	Apache_module	0,02	0,157	0,831	0,164
	1	Definitions	0,18	0,37	1,031	0,191
	1	Definitions_2	0,03	0,261	1,102	0,252
	1	How_Directory.	0,19	0,384	1,182	0,221
	1	http://www1.ists.pwr.wroc.pl/	0,03	0,167	0,891	0,199
	1	StartUp	0,02	0,119	0,651	0,152
	1	Strona_testowa	0,04	0,24	0,971	0,177

Rysunek 5.10: Średni czas odpowiedzi

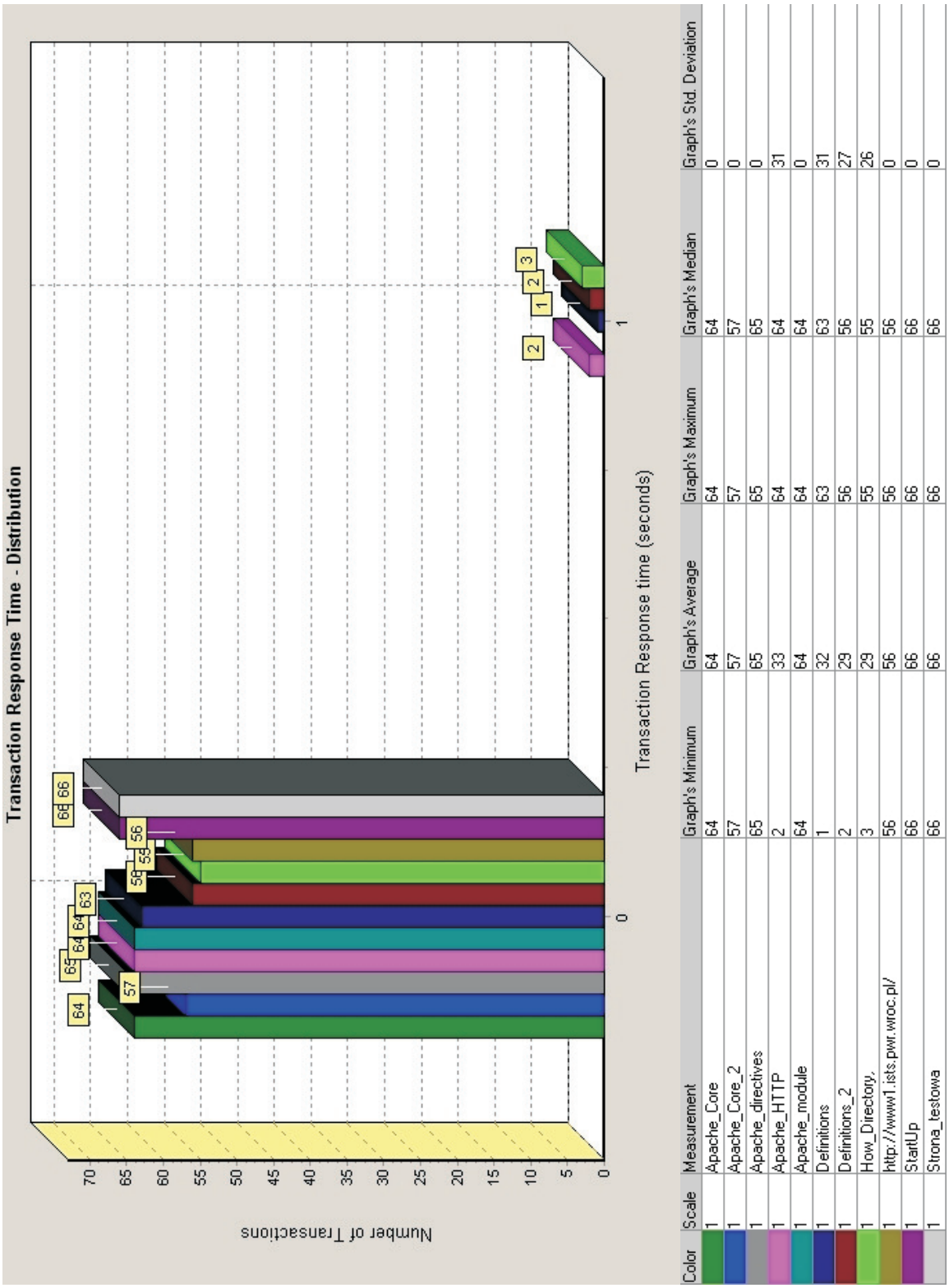


Rysunek 5.11: Zestawienie odpowiedzi w zależności od pobieranej strony



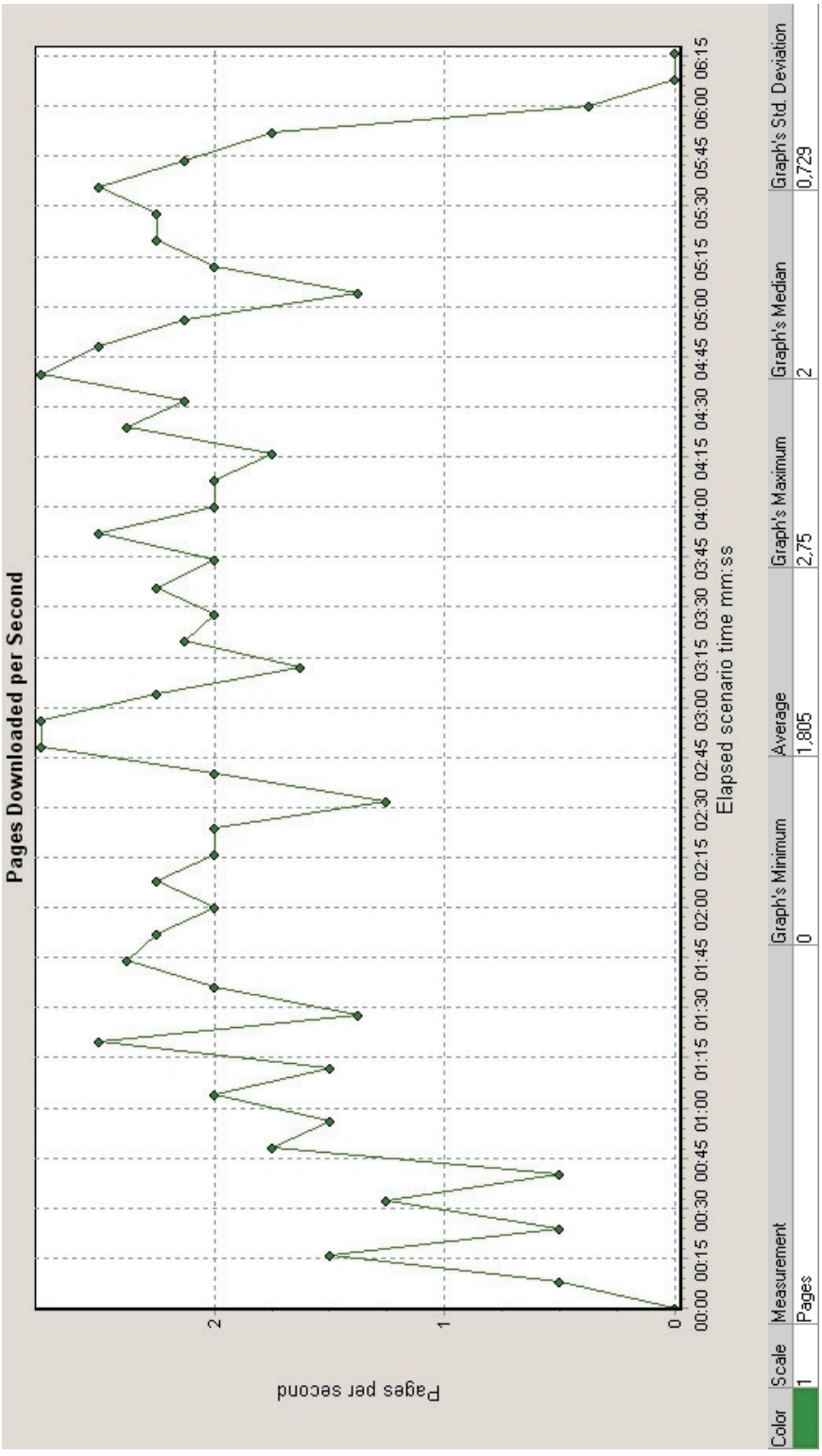
Color	Scale	Measurement	Graph's Minimum	Graph's Average	Graph's Maximum	Graph's Median	Graph's Std. Deviation
	1	StartUp	0,02	0,123	0,65	0,05	0,158
	1	Strona_testowa	0,04	0,246	0,97	0,19	0,186
	1	Apache_HTTP	0,05	0,234	1,03	0,19	0,214
	1	Apache_directives	0,03	0,223	0,85	0,18	0,202
	1	Apache_module	0,02	0,161	0,83	0,14	0,172
	1	Definitions	0,18	0,375	1,03	0,29	0,197
	1	Apache_Core	0,02	0,176	0,62	0,15	0,145
	1	Definitions_2	0,03	0,266	1,1	0,2	0,258
	1	How_Directory.	0,19	0,388	1,18	0,3	0,227
	1	Apache_Core_2	0,04	0,222	0,67	0,2	0,147
	1	http://www1.ists.pwr.wroc.pl/	0,03	0,17	0,89	0,07	0,205

Rysunek 5.12: Czas odpowiedzi transakcji – udział procentowy



Color	Scale	Measurement	Graph's Minimum	Graph's Average	Graph's Maximum	Graph's Median	Graph's Std. Deviation
	1	Apache_Core	64	64	64	64	0
	1	Apache_Core_2	57	57	57	57	0
	1	Apache_directives	65	65	65	65	0
	1	Apache_HTTP	2	33	64	64	31
	1	Apache_module	64	64	64	64	0
	1	Definitions	1	32	63	63	31
	1	Definitions_2	2	29	56	56	27
	1	How_Directory.	3	29	55	55	26
	1	http://www1.ists.pwr.wroc.pl/	56	56	56	56	0
	1	StartUp	66	66	66	66	0
	1	Strona_testowa	66	66	66	66	0

Rysunek 5.13: Czas odpowiedzi transakcji – rozkład



Rysunek 5.14: Liczba stron na sekundę

Spis rysunków