STUDIA I MONOGRAFIE
z. 322

Krzysztof ZATWARNICKI

Systemy webowe z jakością usług uwzględniające kryterium czasowe

PROBLEMY PROJEKTOWANIA

POLITECHNIKA OPOLSKA
ISSN 1429-6063
Opole 2012 ISBN 978-83-62736-75-1

Komitet Redakcyjny Wydawnictw Politechniki Opolskiej
ul. Prószkowska 76

Skład: Oficyna Wydawnicza Politechniki Opolskiej
Nakład: 150 egz. Ark. wyd. 11,0. Ark. druk. 11,0.
Druk i oprawa: Sekcja Poligrafii Politechniki Opolskiej

SPIS TREŚCI

WYKAZ WAŻNIEJSZYCH OZNACZEŃ

a_i, b_i – elementy obciążenia O_i,

A_{ki}, B_{ki} – elementy wektora U_{ki}, wektory parametrów funkcji przynależności wejść,

ar_i^s – liczba równocześnie obsługiwanych żądań HTTP przez s-ty serwer webowy,

arl_i^s – liczba równocześnie obsługiwanych żądań w s-tym serwisie lokalnym,

$ar_{\max}$ – maksymalna dopuszczalna liczba żądań obsługiwanych równocześnie przez realizator,

$ar(\tau)$ – liczba żądań obsługiwanych równocześnie przez realizator w momencie τ,

$ara(\tau)$ – liczba wszystkich żądań znajdujących się w systemie webowym w momencie τ,

$ara_{\max}$ – liczba żądań HTTP w systemie webowym, powyżej której następuje załamanie systemu, tj. czasy obsługi żądań wzrastają powyżej akceptowalnych przez użytkownika wartości,

bd_i^s – liczba żądań HTTP dotyczących obiektów dynamicznych równocześnie obsługiwanych przez s-ty serwer webowy,

btt_i^s – obciążenie sieci rozległej na kierunku od pośrednika do s-tego serwisu lokalnego, czas transferu próbnego obiektu od serwisu lokalnego do pośrednika,

d_i – termin, gdy żądanie powinno zostać przesłane z serwera szeregującego do realizatora,

d_i' – termin, gdy obsługa i-tego żądania HTTP powinna się zakończyć i odpowiedź na żądanie powinno w całości zostać przesłane do przełącznika webowego,

Δd_i – czas przeznaczony na oczekiwanie żądania w kolejce w serwerze pośredniczącym i obsługi żądania w serwerze webowym,

$\Delta d_i'$ – maksymalny czas przeznaczony na obsługę żądania x_i przez serwer webowy,

h_i – element odpowiedzi HTTP y_i, nagłówek odpowiedzi,

h_i' – zmodyfikowany nagłówek odpowiedzi h_i,

i – indeks żądania x_i,

j	–	identyfikator klienta,
j_i	–	identyfikator klienta, który wysłał i -te żądanie,
k_i	–	klasa obiektu żądanego w i -tym żądaniu,
K	–	liczba wyróżnionych klas obiektów,
k_i^l	–	element wektora Z_i , klasa jednego z obiektów nie pobranych jeszcze przez użytkownika w ramach danej strony,
λ	–	współczynnik równoczesności obsługi żądań,
$\mu_{Zal}(a_i)$, $\mu_{Zbm}(b_i)$	–	funkcje przynależności dla zbiorów rozmytych wejść w rozmyto-neuronowym modelu systemu obsługi żądań HTTP,
$\mu_{T_j}(t_i)$	–	funkcje przynależności dla zbiorów rozmytych wyjścia w rozmyto-neuronowym modelu systemu obsługi żądań HTTP,
o_i	–	element odpowiedzi HTTP y_i , żądany w x_i obiekt HTTP,
o_i'	–	zmodyfikowany obiekt o_i , zawierający adresy wskazane w wektorze ou_i' ,
ou_i	–	wektor adresów obiektów zagnieżdżonych lub wskazanych w hiperłączach zawartych w obiekcie o_i ,
ou_i'	–	wektor zmodyfikowanych adresów,
O_i	–	obciążenie systemu obsługi żądań HTTP w i -tym momencie,
O_i^s	–	obciążenie s -tego realizatora (serwera webowego, serwisu lokalnego),
O_i^{sj}	–	obciążenie s -tego serwisu lokalnego, dla j -tego klienta w momencie nadejścia i -tego żądania,
p_i	–	identyfikator strony, do której należy obiekt żądany w i -tym żądaniu,
Q_i	–	kolejka żądań w serwerze szeregującym,
s	–	indeks realizatora (serwera webowego),
S	–	liczba realizatorów (serwerów webowych) wykonujących obsługę żądań HTTP,
so_i	–	numer serwisu lokalnego, przy którym pracuje serwer dystrybucji żądań obsługujący i -te żądanie,
$t_{\max}$	–	zadany nieprzekraczalny czas odpowiedzi dla strony,
$\tilde{t}_i$	–	zmierzony czas odpowiedzi na i -te żądania,
$\tilde{t}_{ig}$	–	zmierzony czas odpowiedzi na żądanie o obiekt o adresie u_{ig}' ,
$\hat{t}_i$	–	szacowany czas odpowiedzi na i -te żądanie,

$\hat{t}_i^s$ – szacowany czas odpowiedzi dla s-tego serwera na i-te żądanie HTTP,

$\hat{t}_{ig}^s$ – szacowany czas odpowiedzi dla s-tego serwera na żądanie o obiekt o adresie u'_{ig},

$\hat{t}'_{ki}$ – szacowany czas odpowiedzi na i-te żądanie obsługiwane przez realizator,

$\bar{t}_i^{sj}$ – obciążenie sieci Internet na kierunku od j-tego klienta do s-tego serwisu lokalnego,

$\dot{t}_{p_i}$ – zmierzony czas odpowiedzi strony o identyfikatorze p_i,

$\tau_i^{(1)}$ – moment nadejścia i-tego żądania do serwisu,

$\tau_i^{(2)}$ – moment, gdy i-te żądanie opuszcza kolejkę Q_i,

tp_{p_i} – czas liczony od momentu nadejścia pierwszego żądania o obiekt strony p_i do momentu nadejścia i-tego żądania,

T_{ki} – element wektora U_{ki}, wektor parametrów funkcji przynależności wyjścia t,

u_i – adres obiektu żądanego w i-tym żądaniu,

u_{ig} – adres obiektu zagnieżdżonego w obiekcie o_i lub hiperłączu do innego obiektu udostępnianego w serwisie webowym,

u'_{ig} – zmodyfikowany adres, adres obiektu wskazujący ten sam obiekt, co w u_{ig} w innym serwisie lokalnym niż w oryginalnym adresie,

U_i – macierz parametrów modelu systemu obsługi żądań HTTP w momencie nadejścia i-tego żądania,

U_{ki} – element wektora U_i, wektor parametrów modelu systemu obsługi żądań HTTP dla obiektów k-tej klasy,

U'_i – parametry realizatora w modelu realizatora systemu WEDF,

w_i – decyzja, numer realizatora (serwera) wybranego do obsługi i-tego żądania,

x_i – i-te żądanie HTTP,

X – zbiór żądań HTTP obsługiwanych prawidłowo w serwisie webowym,

y_i – odpowiedź HTTP na i-te żądanie HTTP,

y_i^s – odpowiedź s-tego realizatora (serwera webowego) na i-te żądanie HTTP,

y'_i – zmodyfikowana odpowiedź y_i na i-te żądanie,

Z_i – wektor klas obiektów należących do strony, a nie pobranych jeszcze w ramach danej strony przez klienta,

WYKAZ WAŻNIEJSZYCH AKRONIMÓW

2D	–	ang. Session-based two-dimensional service
ADNS	–	ang. Authoritative Domain Name Server
CAP	–	ang. Client-Aware Policy
CDN	–	ang. Content Delivery Network
CDNS	–	ang. Cluster DNS
DNS	–	ang. Domain Name System
DNSLB	–	ang. DNS-Based Load Balancing
DS	–	ang. Direct Service
DWFS	–	ang. Dynamic Weighted Fair Sharing
EDF	–	ang. Earliest Deadline First
EORMP	–	ang. E-commerce Oriented Resource Policy
FARD	–	ang. Fuzzy Adaptive Request Distribution
FCF	–	ang. Fastest Connection First
FCFS	–	ang. First Come First Served
FIFO	–	ang. First In First Out
FNRD	–	ang. Fuzzy Neural Request Distribution
GARD	–	ang. Global Adaptive Request Distribution
GARDiB	–	ang. Global Adaptive Request Distribution with Broker
GGARDiB	–	ang. Guaranteed Global Adaptive Request Distribution with Broker
HTML	–	ang. HyperText Markup Language
HTTP	–	ang. Hypertext Transfer Protocol
IP	–	ang. Internet Protocol
KARO	–	ang. Key customers And Revenue Oriented admission and scheduling
LARD	–	ang. Locality-Aware Request Distribution
LFNRD	–	ang. Local Fuzzy Neural Request Distribution
LL	–	ang. Least Loaded
LMAC	–	ang. Latency Targeted Multi-Class Admission Control
LRU	–	ang. Last Recently Used

LVS – ang. Linux Virtual Server
MLF – ang. Most Loaded First
QoWS – ang. Quality of Web Service
RR – ang. Round-Robin
RTT – ang. Round Trip Time
SALSA – ang. Simulated Annealing Load Spreading Algorithm
SED – ang. Shortest Expected Delay
SI – ang. Session Identifiers
SJF – ang. Shortest Job First
SOC – ang. Self Overload Control
SP – ang. Service Partitioning
SRPT – ang. Shortest Remaining Processing Time
SRRT – ang. Shortest Remaining Response Time
STFC – ang. Self-tuning Fuzzy Contoroller
TCP – ang. Transmission Control Protocol
URI – ang. Uniform Resource Identifier
W3C – ang. World Wide Web Consortium
WEDF – ang. Web Earliest Deadline First
WRR – ang. Weighted Round-Robin
WRR_L – ang. Weighted Round-Robin Load
WRR_T – ang. Weighted Round-Robin Transfer
WWW – ang. Word Wide Web

1. WSTĘP

Internet jest medium, dzięki któremu prowadzone mogą być zróżnicowane aktywności człowieka takie jak: zabawa, rozrywka, pozyskiwanie wiedzy lub prowadzenie biznesu. Podstawową usługą działającą w Internecie jest usługa WWW (ang. Word Wide Web). WWW lub też po prostu Web jest przestrzenią informacji dostępnych poprzez sieć komputerową [21]. Usługa WWW zyskała ogromną popularność, dzięki zastosowaniu prostego graficznego interfejsu pozwalającego użytkownikom na przeglądanie stron WWW oraz przemieszczanie się między nimi poprzez kliknięcia na odnośniki nazywane hiperłączami [103]. Podstawowymi semantycznymi komponentami tworzącymi usługę WWW są: hipertekstowy język znaczników HTML (ang. HyperText Markup Language) [88], protokół przesyłania dokumentów hipertekstowych HTTP (ang. Hypertext Transfer Protocol) oraz uniwersalne identyfikatory zasobów URI (ang. Uniform Resource Identyfier) [79].

Źródłami stron HTML pobieranymi przez klientów są serwisy WWW nazywane również serwisami webowymi. Strony HTML przesyłane są z wykorzystaniem protokołu HTTP (ang. Hypertext Transfer Protocol) z serwisów WWW do klientów, najczęściej poprzez sieć rozległą Internet. Interakcja klienta z serwisem WWW polega na wysyłaniu żądań HTTP od klienta do serwisu WWW oraz przesyłaniu odpowiedzi HTTP w postaci obiektów HTTP z serwisu WWW do klienta. Żądania w serwisie webowym obsługiwane są przez serwery WWW nazywane również serwerami webowymi. Serwis WWW wraz z urządzeniami umożliwiającymi dostarczanie żądanych obiektów HTTP tworzy system webowy.

Jakość działania serwisów webowych (ang. Quality of Web Service) może być oceniana przez użytkowników na wiele różnych sposobów. Z jednej strony, interesujące treści prezentowane w serwisie zapewniają wzrost liczby użytkowników, z drugiej jednak strony ważne jest, aby treści dostarczane były w określonym czasie po ich zażądaniu. Użytkownicy oceniać będą jakość serwisu jako niską, gdy czas potrzebny do pozyskania nawet najciekawszych treści będzie zbyt długi. Dlatego też jakość serwisów webowych lub też całych systemów webowych oceniana jest często głównie w kategorii czasu uzyskania dostępu do żądanych obiektów HTTP lub całych stron [25, 34, 122, 212].

Opóźnienia dotyczące pozyskiwania żądanych treści związane są głównie z opóźnieniami występującymi przy przesyłaniu danych przez sieć rozległą Internet oraz opóźnieniami występującymi przy przetwarzaniu żądań HTTP w ośrodkach webowych.

Skuteczną metodą zmniejszania opóźnień związanych z przesyłaniem danych jest polepszenie warunków transmisji np. poprzez powiększenie przepustowości łącza bądź uproszczenie architektury sieci i zmianę technologii przesyłu danych [68]. Jednak takie zabiegi nie są możliwe we wszystkich przypadkach, a zwłaszcza tam, gdzie źródło opóźnień związane jest z działaniem samej sieci rozległej, na której budowę i działanie nie mają bezpośredniego wpływu projektanci serwisów webowych. Inne techniki pozwalające na zmniejszenie opóźnień związanych z przesyłaniem danych polegają na „przybliżeniu" (w sensie sieciowym) żądanych obiektów do klientów. Jedną z podstawowych technik tu stosowanych jest wykorzystanie podręcznych pamięci przeglądarek internetowych, w których przechowywane są pobrane wcześniej obiekty [53, 189].

Jedną z najstarszych technik przybliżania danych do klienta jest stosowanie serwerów typu Mirror, które umożliwiają pobranie danych ze źródła najbliższego klientowi. Wadą tego rozwiązania jest brak transparentności dla użytkownika i zmuszanie użytkownika do świadomego wyboru odpowiedniego serwera typu Mirror [68].

Inną często stosowaną techniką jest wykorzystanie serwerów typu *caching proxy*, znajdujących się w niewielkiej odległości od klientów. Serwery te w swych pamięciach podręcznych przechowują aktualne obiekty HTTP pobierane przez klientów. Tego typu serwery pośredniczące są najczęściej umieszczane na krawędzi sieci lokalnej. Innym sposobem wykorzystania serwerów typu *cache* jest budowa Sieci Dystrybucji Treści (CDN, ang. Content Delivery Network). W rozwiązaniu tym odpowiednie serwery typu *cache* rozmieszczone w różnych częściach sieci rozległej Internet dostarczają treści klientom znajdującym się w bliskiej od nich odległości [49, 136].

Stosowanie serwerów typu *cache* ma jednak istotne ograniczenia. Spotykane obecnie w Internecie serwisy webowe udostępniają nie tylko obiekty statyczne, czyli pliki przechowywane na dyskach serwerów WWW, ale również dostarczają obiekty dynamiczne, tworzone na bieżąco po otrzymaniu żądania HTTP. Tworzone dynamicznie obiekty HTTP dla takiego samego żądania mogą być za każdym razem inne. Obiektów dynamicznych nie można przechowywać w serwerach typu *cache*. Dlatego też często stosowane są systemy globalnie rozmieszczone, których poszczególne elementy infrastruktury umieszczane są w różnych oddalonych od siebie rejonach świata.

Opóźnienia dotyczące pozyskiwania żądanych treści związane mogą być również z czasem przetwarzania żądań HTTP w ośrodkach webowych. Przeprowadzone badania [53, 92] wskazują, że średnio 40% opóźnień związanych z pozyskiwaniem danych żądanych przez klientów wynika z niesprawnego przetwarzania żądań HTTP przez serwisy WWW. Z innych badań [13, 15, 86] natomiast wynika, że opóźnienia powstałe na obciążonym serwisie webowym mogą stanowić nawet ponad 80% czasu odpowiedzi dla plików małych i 50% dla plików średniej wielkości. Obecnie również na czas obsługi żądań wpływa znacząco czas przygotowania obiektów dynamicznych, których zawartość często

tworzona jest w oparciu o dane pochodzące z baz danych. Dlatego też badania nad wydajnością serwisów webowych mają duże znaczenie.

Istnieje wiele różnych sposobów umożliwiających zmniejszenie opóźnień związanych z przetwarzaniem żądań HTTP. Jednym z najczęściej stosowanych sposobów, posiadającym niestety liczne ograniczenia, jest wykorzystanie wydajniejszego serwera WWW i serwerów biorących udział w przetwarzaniu [53]. Rozwiązanie to nie daje jednak dużych możliwości w skalowaniu systemu webowego. Inną metodą jest zastosowanie serwerów typu *reverse proxy*. Serwery te posiadają, podobnie jak serwery *proxy*, pamięć podręczną do przechowywania statycznych obiektów HTTP, ale w odróżnieniu od serwerów *proxy* umieszczane są w bliskiej odległości od serwisu webowego. Serwery tego typu nie przyspieszają obsługi żądań dotyczących obiektów dynamicznych. Dlatego też w przypadku systemów webowych, które powinny cechować się skalowalnością oraz dużą wydajnością w obsłudze żądań również dynamicznych, najczęściej stosuje się klastry serwerów lokalnie rozmieszczonych lub też farmy serwerów. W obu rozwiązaniach stosuje się kilka serwerów WWW oraz innych serwerów, tworzących infrastrukturę systemu webowego, połączonych wysoko wydajną siecią lokalną, przy czym klastry serwerów lokalnie rozmieszczonych posiadają jeden punkt dostępu do usług (jeden wirtualny adres IP), natomiast farmy serwerów kilka takich punktów [81]. Zagadnienia dystrybucji żądań w lokalnych klastrach serwerów webowych rozpatrywane były w licznych publikacjach, co świadczyć może o ważności zagadnienia i dużych możliwościach w podnoszeniu jakości usług webowych [12, 23–25, 52, 53, 56, 60, 62, 63, 77, 81, 110, 111, 124, 131, 133, 135, 139, 148, 152, 153, 155, 156, 159, 161, 167, 190, 215, 218, 224–226].

Innym sposobem podnoszenia wydajności serwisu WWW, wcześniej już wspomnianym, jest zastosowanie systemów globalnie rozmieszczonych. W systemach tych, w różnych lokalizacjach geograficznych mogą być umieszczone nie tylko pojedyncze serwery webowe, ale całe klastry serwerów tworząc systemy globalnie rozmieszczonych klastrów webowych. W bogatej literaturze przedmiotu dotyczącej rozwiązań wdrożonych oraz projektowania systemów globalnie rozproszonych proponowane są liczne rozwiązania w tym zakresie [7, 16, 34, 54, 57, 66, 67, 76, 95, 127, 129, 134, 137, 145, 146, 162, 157, 180, 222].

Wskazane powyżej podstawowe metody mają wpływ na opóźnienia przesyłania danych przez sieć oraz opóźnienia przetwarzania żądań HTTP w systemach webowych. Stosowanie odpowiednich metod ma więc wpływ na jakość usług serwisów webowych. Dla celów pracy przyjęte zostaje następujące stwierdzenie:

System webowy z jakością usług jest to system webowy, w którym wykorzystywane są odpowiednie metody umożliwiające osiąganie zamierzonych celów w zakresie jakości oferowanych usług.

Dotychczas wiele systemów webowych, a zwłaszcza klastrowych serwerów webowych, było konstruowane tak, aby równoważyć obciążenia [56, 60] lub maksymalizować przepustowość serwisu [12, 135, 225], co jest uzasadnione

z punktu widzenia usługodawcy dostarczającego treści webowe. Obecnie jakość usług serwisów jest oceniana przez użytkowników, którzy bezpośrednio wpływają na popularność i zasadność funkcjonowania serwisów. Dlatego też w tym opracowaniu pisząc o jakości usług, będziemy mieli na myśli czas ich realizacji.

Zagadnienia dotyczące podnoszenia oraz gwarantowania jakości usług systemów webowych związane są mocno z zagadnieniami ogólnej teorii systemów, metodologii podejmowania decyzji oraz sterowania [22, 48, 164]. W konstruowaniu metod zarządzania żądaniami HTTP korzysta się często z klasycznych podejść znanych z teorii szeregowania i alokacji zadań w systemach rozproszonych, a także z rezultatów uzyskanych przy rozwiązywaniu problemów alokacji zadań i rozdziału zasobów w systemach maszyn równoległych oraz systemach scentralizowanych. Warto tu wspomnieć dla przykładu o aktywnie rozwijanych podejściach do obsługi żądań HTTP przez pojedynczy serwer WWW, w których proponowane są efektywniejsze w stosunku do klasycznej polityki FIFO (ang. First In First Out) algorytmy kolejkowania żądań przez serwery WWW [17, 87, 155].

Szczególną rolę w opracowywaniu metod podejmowania decyzji w systemach webowych mogą odegrać metody opatrzone wspólną nazwą sztucznej inteligencji. Stosowanie tego typu metod wiąże się zwykle z nietradycyjnym opisem obiektu podejmowania decyzji i algorytmu decyzyjnego. Spośród wielu metod w tym zakresie należy wymienić dwie, które będą zastosowane w niniejszej pracy, a mianowicie metody wykorzystujące sztuczne sieci neuronowe oraz opis rozmyty. Potencjał obu metod może być wykorzystany w opisie niepewności oraz opracowaniu określonej koncepcji adaptacji. Opis rozmyty jest uznanym sposobem modelowania w sytuacji, gdy dysponujemy bardzo ograniczoną informacją o obiekcie [73]. Dokładny opis tych metod i cel ich stosowania w systemach webowych można znaleźć w [148, 172, 173, 195, 206, 210, 218].

Specyfika problemu podejmowania decyzji w sterowaniu ruchem żądań HTTP i obsługi żądań w ośrodkach webowych wiąże się z ograniczeniami dotyczącymi przyjętych standardów pracy w sieci Internet, protokołów przesyłania danych we wszystkich warstwach modelu TCP/IP [46, 82, 143, 147, 166], implementacji w węzłach sieci stosów protokołowych, sposobu obsługi żądań HTTP w ośrodkach webowych, w tym w szczególności w serwerach WWW [21, 78], serwerach aplikacji i bazodanowych oraz z charakterystykami pracy klientów. Wprowadzenie nowego rozwiązania dotyczącego obsługi żądań HTTP w serwisach webowych nie może zaburzyć obowiązujących zasad działania Internetu i warunków współpracy z siecią WWW, które wypracowywane są m.in. przez konsorcjum W3C [171]. Każde z nowych rozwiązań musi charakteryzować się własnościami pracy w czasie rzeczywistym. Specyfika przetwarzania żądań HTTP w serwisach webowych przejawia się między innymi w: rozkładach opisujących proces napływu żądań HTTP od ośrodka przetwarzania [13–15, 177], rozkładach czasów przebywania i obsługi w systemach komputerowych [12, 217] oraz czasów przesyłania danych przez sieć, zmienności tych

charakterystyk związanych np. z porą dnia, tygodnia, miesiąca, roku i trudnymi do przewidzenia wydarzeniami społecznymi, niepewności w opisie zadania obliczeniowego serwisu oraz podejmowanej decyzji [33, 41, 118–120, 214].

Projektowanie systemów webowych, w których znaczenie ma jakość usług, wiąże się często z odpowiedzią na pytanie, jaki cel ma zostać osiągnięty w zakresie jakości usług w procesie podejmowania decyzji w projektowanym systemie. Najczęściej wyróżniane są dwa cele [47]: ekstremalizacja wybranego wskaźnika jakości np. minimalizacja czasu odpowiedzi lub też takie podejmowanie decyzji, aby jakość usług utrzymana była na zadanym poziomie. W pracy [121] przyjęte zostało, że usługi dostępne w sieciach komputerowych mogą zostać podzielone na trzy typy ze względu na jakość usług:

- usługi typu best-effort (jakość najlepsza z możliwych) – usługi tego typu są realizowane w ramach dostępnych do realizacji środków, najczęściej odpowiednie podsystemy wykonują usługę z najwyższą jakością w ramach dostępnych środków, ale nie ma żadnych gwarancji, że jakość usług będzie wysoka oraz że usługa w ogóle zostanie zrealizowana,
- usługi przewidywalne – oferują jakość na określonym, zamkniętym w pewnych granicach poziomie. Usługi te muszą być konfigurowalne, tzn. musi istnieć możliwość zmiany poziomu jakości usług oraz możliwość zmierzenia jakości i jej weryfikacji. W usługach przewidywalnych dopuszczalne jest przekraczania przyjętych granic jakości usług, o ile tendencja do przekraczania granic nie jest stała,
- usługi gwarantowane – podobnie jak usługi przewidywalne oferują jakość na wymaganym, zamkniętym w ramach pewnych granicach poziomie. W przeciwieństwie do usług przewidywalnych, w usługach gwarantowanych nie jest dopuszczalne przekraczanie przyjętych granic wartości jakości usług, ponieważ wiąże się to najczęściej z poważnymi konsekwencjami finansowymi.

W ramach usług typu best-effort celem podejmowania decyzji jest najczęściej maksymalizacja jakości usług. W usługach przewidywalnych i gwarantowanych celem sterowania jest utrzymanie jakości usług na wymaganym poziomie. Systemy oferujące jakość usług przewidywalną lub gwarantowaną są systemami z zadanym wyjściem.

Przyjmując, że jakość usług związana jest bezpośrednio z czasem odpowiedzi na żądania klienta, możemy powiedzieć, że obecnie w sieci rozległej Internet prawie wszystkie usługi, w tym również usługi oferowane przez systemy webowe są usługami typu best-effort. Obecnie stosowane protokoły sieciowe, poczynając od protokołów warstwy łącza danych, a kończąc na protokołach warstwy aplikacji, najczęściej nie dają możliwości zarządzania jakością usług. Dodatkowo budowa i działanie aplikacji realizujących usługi np. serwery WWW Apache i IIS, serwery poczty Sendmail, Quickmail i inne nie dają kontroli nad jakością usług. Innym czynnikiem wpływającym na brak możliwości zarządzania jakością usług jest to, że poszczególne części sieci Internet są zarządzane przez różnych, bardzo licznych właścicieli, którzy realizują różne cele

strategiczne. Dlatego też, o ile możliwe jest wdrażanie w sieci Internet usług przewidywalnych, to już niemożliwe jest na dzień dzisiejszy wdrażanie usług gwarantowanych (co nie oznacza, że nie jest to możliwe w prywatnych sieciach rozległych). Usługi gwarantowane wymagają, aby jakość usług była gwarantowana od początkowego elementu w sieci poprzez wszystkie pośredniczące aż do końcowego, co jest nie do spełnienia w przypadku sieci Internet, która daje usługi typu best-effort.

Przez wiele lat w większości opracowań naukowych dotyczących jakości usług webowych opisywane były zagadnienia podnoszenia jakości usług w systemach webowych. Jednak wraz z rozwojem usług w Internecie, w tym głównie usług biznesowych pojawiła się potrzeba realizacji usług na wymaganym poziomie dającym pewność klientom obsługi dobrej jakości. Dlatego też od początku nowego wieku zaczęły pojawiać się, a obecnie stanowią szeroki nurt, publikacje dotyczące systemów webowych realizujących usługi przewidywalne [1, 4, 6, 23, 40, 51, 55, 58, 61, 87, 128, 154, 160, 228]. W większości opracowań nie wyróżnia się jednak osobnych grup systemów realizujących usługi przewidywalne i gwarantowane określając obie grupy mianem systemów gwarantowanych [1, 172]. W tym opracowaniu dla grupy systemów oferujących usługi gwarantowane i przewidywalne użyte zostanie wspólne określenie: system z zadaną jakością usług. Pisząc jednak o systemach określanych przez ich twórców jako systemy gwarantujące jakość usług, podtrzymana zostanie nomenklatura wprowadzona przez twórców.

Zagadnieniom projektowania systemów webowych z jakością usług poświęconych zostało wiele publikacji, między innymi [1, 7, 40, 56, 58, 173, 225]. Wśród wielu spośród wymienionych tu prac zaproponowane zostały całkowicie nowatorskie podejścia i metody w dziedzinie systemów webowych. Ze względu na ogromny rozwój usług internetowych istnieje obecnie stałe zapotrzebowanie na opracowywanie nowych metod w zakresie podnoszenia i gwarantowania jakości. Monografia ta poświęcona została zagadnieniom projektowania systemów webowych z jakością usług.

Projektowanie wydajnych systemów webowych z jakością usług nie jest zadaniem prostym, głównie ze względu na trudności w podejmowaniu decyzji w warunkach niepewności i zmienności charakterystyki ruchu i obiektu. Wykonanie projektu wymaga opracowania odpowiednich algorytmów podejmowania decyzji, w tym algorytmów dystrybucji lub też szeregowania żądań HTTP oraz metod umożliwiających osiąganie założonego w projekcie celu. Pojęcie szeregowania żądań jest różnie definiowanie w poszczególnych opracowaniach naukowych. W ramach tego opracowania przyjmuje się, że szeregowanie oznacza ustalenie kolejności obsługi żądań, natomiast dystrybucja wyznaczenie realizatora oraz przekazanie do niego żądania.

W procesie projektowania systemów można wyróżnić następujące fazy [47]:

1. analiza systemowa obiektu,
2. ustalenie modelu obiektu,

3. wyznaczenie algorytmów podejmowania decyzji,
4. dobór struktury realizującej algorytmy,
5. weryfikacja i korekty.

Analiza systemowa obiektu polega na ustaleniu, jaki obiekt lub proces będzie podlegał procesowi zarządzania, ustaleniu wielkości charakterystycznych takich jak zakres podejmowanych decyzji oraz wielkości wejściowe. Jednym z ważniejszych punktów tego etapu jest wskazanie celów działania projektowanego systemu.

Ustalenie modelu obiektu w projektowanym systemie podnosi jakość podejmowanych decyzji. Jednakże tylko w nielicznych systemach webowych takie modele są wykorzystywane w procesach podejmowania decyzji [28, 173, 212, 219, 221], przeważnie decyzje podejmowane są z wykorzystaniem prostych algorytmów heurystycznych, takich jak algorytm karuzelowy lub jego odmiany. Opracowanie odpowiednich modeli dla systemów webowych jest zagadnieniem bardzo złożonym ze względu na wspomniane już wcześniej niepewność i zmienność charakterystyki ruchu i obiektu, dodatkowym problemem jest wydajność systemów podejmowania decyzji. Jednak odpowiednio dobrany model obiektu daje możliwość przewidywania skutków podjętych decyzji, a co za tym idzie, podejmowanie decyzji optymalnych lub paraoptymalnych lub też umożliwia tworzenie systemów, których celów sterowania nie dałoby się osiągnąć bez wykorzystania modeli. Odpowiednio opracowany uniwersalny model elementów systemu webowego zostanie przedstawiony w dalszej części pracy.

Kolejna faza w projektowaniu polega na opracowaniu odpowiedniego algorytmu podejmowania decyzji lub też zestawu powiązanych ze sobą algorytmów składowych. Działanie takich algorytmów powinno być ściśle związane ze wskazanym wcześniej celem działania systemu. Wykorzystywane obecnie systemy webowe stosują często proste algorytmy cechujące się krótkim czasem podejmowania decyzji, jednak jak pokazały prace [26, 32] warto przeznaczyć więcej czasu na podjęcie decyzji, aby otrzymać znacząco lepszy wynik decyzji. W przeglądowej publikacji dotyczącej klastrowych systemów webowych [53] wskazana została potrzeba opracowania odpowiednich adaptacyjnych algorytmów podejmowania decyzji w systemach webowych z jakością usług, ze względu na ich możliwości dopasowania się do zmieniających się warunków pracy w Internecie i w ośrodkach webowych. Prezentowane w dalszej części tego opracowania projekty systemów webowych wykorzystywać będą właśnie algorytmy adaptacyjne.

Kolejna faza w projektowaniu polega na wskazaniu odpowiednich środków technicznych umożliwiających realizację projektu w praktyce. W przypadku systemów webowych należy wyznaczyć odpowiednie metody określające sposób działania systemu webowego oraz przedstawić projekty urządzeń (lub oprogramowania) sterujących systemem webowym zgodnie z przyjętymi metodami. Opracowując metody funkcjonowania systemów webowych należy wskazać odpowiednie rozwiązania techniczne na poziomie protokołów sieciowych, a zwłaszcza protokołów HTTP, TCP oraz IP. Uwzględnienie sposobu działania

protokołów sieciowych, programów, urządzeń oraz różnorodność stosowanych architektur i implementacji nakłada bardzo duże ograniczenia na możliwości w projektowaniu systemów webowych z jakością usług i wymaga od projektanta bardzo specjalistycznej wiedzy. Wspomniane ograniczenia należy uwzględniać w każdej fazie projektowania.

Ostatnią z wymienionych faz projektowania systemów jest weryfikacja i korekty. Obie te czynności przeprowadzane są na każdym etapie projektowania. Jednym z ważniejszych zadań omawianej fazy jest przeprowadzenie badań umożliwiających ocenę proponowanych rozwiązań, określającą przydatność i zasadność realizacji i wdrożenia projektu. Często stosowanym narzędziem analizy jest symulacja komputerowa [47], ale również są stosowane odpowiednie metody analizy teoretycznej z wykorzystaniem modelowania matematycznego oraz prowadzone są badania z wykorzystaniem systemów prototypowych. Bogata literatura przedmiotu w zakresie wykorzystania metod analitycznych wskazuje na szerokie możliwości oraz skuteczność takich podejść. Metoda ta jest dla przykładu z powodzeniem stosowana dla algorytmów rozdziału zasobów i alokacji zadań w systemach wieloprocesorowych np. [22, 97, 98]. Należy jednak zauważyć, że przyjmowane w badaniach modele systemów są zazwyczaj bardzo uproszczone, ponieważ tylko dla takich modeli udaje się skutecznie stosować metody analityczne. W literaturze dotyczącej projektowania systemów webowych z jakością usług, poza nielicznymi wyjątkami [72, 86, 102, 126, 223], rzadko spotyka się analityczne podejście do oceny stosowanych metod. Trudności wynikają z dużej złożoności problemu modelowania zachowań systemów webowych.

Metodami, które są najczęściej stosowane w ocenie systemów webowych, są metody symulacyjne [7, 28, 52, 72, 168, 212, 213]. Zaletą stosowania metod symulacyjnych jest fakt, że budowane modele symulacyjne mogą mieć dowolną dokładność, w zależności od wymogów stawianych w prowadzonych badaniach. Dlatego też całe duże fragmenty systemów mogą być modelowane jako jeden uproszczony moduł lub też możliwe jest modelowanie zachowania konkretnych algorytmów przy uwzględnieniu sposobów ich implementacji. Inną zaletą stosowania metod symulacyjnych jest możliwość przebadania dużej liczby różnych konfiguracji sprzętowych, które mogą być spotykane w rozwiązaniach praktycznych. Duże znaczenia w modelowaniu systemów webowych ma zastosowanie odpowiednio skonstruowanych generatorów żądań HTTP, które generowałyby strumienie żądań charakteryzujące się cechami strumieni żądań obserwowalnych w Internecie. Stosowane obecnie w badaniach symulacyjnych generatory żądań wykorzystują logi serwerów webowych symulując postępowanie rzeczywistych klientów [1, 26, 148] lub też symulowany ruch w sieci generowany jest zgodnie z wyznaczonymi rozkładami prawdopodobieństwa, uzyskanymi na podstawie przeprowadzonych obserwacji w sieci [15, 17, 71, 201, 203, 219]. Przekrojowe badania symulacyjne są zazwyczaj prowadzone przed podjęciem decyzji o wdrożeniu danego systemu webowego i stanowią często integralną część procesu wdrożeniowego.

Przed wdrożeniem systemów webowych konieczne jest zbudowanie odpowiednich prototypów, które po przeprowadzeniu badań i odpowiednich zmianach staną się w pełni funkcjonalnymi działającymi systemami. Budowa systemów prototypowych, a zwłaszcza ich weryfikacja na odpowiednim sprzęcie komputerowym w środowisku Internetu jest zazwyczaj bardzo kosztowna. Wiąże się to z wykonaniem oprogramowania, często zakupem specjalistycznych urządzeń sieciowych, odpowiednich serwerów oraz oprogramowania. W przypadku budowy systemów webowych globalnie rozmieszczonych znacząca część kosztów związana jest z umieszczeniem poszczególnych części infrastruktury w różnych globalnie rozmieszczonych ośrodkach. Badania nad systemami webowymi prowadzone z wykorzystaniem rozwiązań prototypowych opisane są w [25, 80, 87].

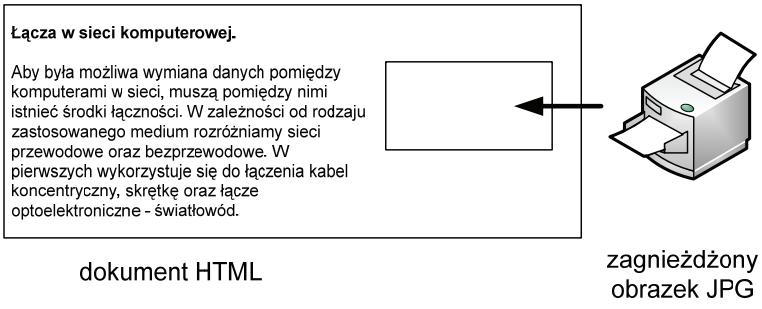
Badania prowadzone z wykorzystaniem metod symulacyjnych i rozwiązań prototypowych nie dają jednoznacznych odpowiedzi, czy zastosowane metody spełniają wymagania i są lepsze od innych metod we wszystkich przypadkach [142]. Jednak stosowanie odpowiednich programów symulacyjnych dobrze odwzorowujących pracę poszczególnych elementów badanego systemu pozwala przypuszczać, że odpowiednie własności wskazane po wykonaniu badań dla danego układu lub systemu posiadać będzie cała klasa podobnych rozwiązań.

Ze względu na wskazane własności poszczególnych metod badawczych przyjęte zostało, że przedstawione w pracy rozwiązania dotyczące systemów webowych z jakością usług weryfikowane będą z wykorzystaniem ujednoliconej platformy symulacyjnej umożliwiającej prowadzenie badań symulacyjnych zarówno dla systemów webowych jednoserwerowych, klastrów serwerów lokalnie rozmieszczonych oraz klastrów globalnie rozmieszczonych w Internecie. Dzięki takiemu podejściu możliwe będzie porównanie wyników badań uzyskanych dla wskazanych powyżej rozwiązań.

W celu wprowadzenia w tematykę zagadnienia, ujednolicenia oraz zdefiniowania pojęć w rozdz. 1.1 opisane zostaną podstawowe zagadnienia dotyczące protokołu HTTP oraz jakości usług webowych. Następnie w rozdz.1.2 przedstawiona zostanie klasyfikacja systemów webowych z jakością usług, a w rozdz. 1.3 wskazane zostanie miejsce prezentowanych w pracy rozwiązań wśród innych znanych rozwiązań. Rozdział 1.4 zawiera cel pracy oraz przegląd treści.

1.1. OPIS USŁUGI WWW

Dokumenty dostępne poprzez usługę WWW tworzone są w hipertekstowym języku znaczników HTML. Strona WWW, nazywana również dalej stroną internetową lub webową, zawiera dokument HTML będący szkieletem strony, posiadającym treść oraz obiekty zagnieżdżone w stronie. Obiektami zagnieżdżonymi mogą być obrazki (np. JPG, PNG, BMP), obiekty wideo, programy (np. flash, Java Script) oraz inne obiekty zawierające informacje sterujące sposobem prezentacji strony (pliki stylów). Na rys. 1.1 przedstawiona została budowa przykładowej strony HTML.



Rys. 1.1. Budowa strony HTML

Do pobierania stron z serwisów webowych jest wykorzystywany, jak zostało to już wspomniane, protokół HTTP. HTTP jest protokołem warstwy aplikacji w modelu ISO/OSI [96] i definiują go między innymi dokumenty RFC 2068 oraz 2616 [78, 79]. Protokół HTTP określa interakcję pomiędzy klientem a serwisem webowym, na którą składają się żądania HTTP wysyłane przez klientów oraz odpowiedzi HTTP zawierające żądane obiekty HTTP, wysyłane przez serwis webowy. Obiekty HTTP są dokumentami HTML oraz pozostałymi obiektami żądanymi przez klienta.

Obecnie wykorzystywane są dwie wersje protokołu HTTP, są to: HTTP 1.0 [20] oraz HTTP 1.1 [79]. Występuje jedna zasadnicza różnica w obu protokołach mająca duży wpływ na jakość usług. Polega ona na wprowadzeniu w protokole HTTP 1.1, ustawionej jako domyślna, opcji trwałych połączeń (ang. persistent connection lub keep-alive), dzięki której w ramach jednego połączenia TCP klienta z serwerem możliwe jest pobranie wielu obiektów HTTP. Ponieważ obecnie wszystkie nowe przeglądarki domyślnie pracują w protokole HTTP 1.1, dlatego też w dalszej części opracowania skrót HTTP oznaczał będzie protokół w wersji 1.1.

Protokół HTTP jest protokołem bezstanowym, w którym poszczególne żądania HTTP nie są ze sobą związane, dlatego też w przypadku protokołu HTTP nie można mówić o sesjach.

Żądanie HTTP w protokole HTTP 1.1 wysyłane przez klienta składa się z następujących ważniejszych elementów:

- typ żądania określający akcję, którą serwer WWW powinien podjąć po otrzymaniu żądania, do najczęściej spotykanych typów należą GET oraz POST,
- adres URI żądanego obiektu,
- nazwa serwisu, do którego żądanie jest skierowane,
- pole *cookie* niosące informacje dotyczące klienta, informacje te mogą zostać ustawione przez serwis WWW,
- pozostałe informacje dotyczące typu przyjmowanych dokumentów, autoryzacji, trwałych połączeń i innych.

Odpowiedź HTTP składa się z następujących ważniejszych elementów:

- nagłówek odpowiedzi zawierający: numer błędu odpowiedzi, typ obiektu HTTP, wielkość obiektu, może zawierać żądanie ustawienia wartości pewnych zmiennych w polu cookie. Średnia długość nagłówka wynosi 290 B [126],
- żądany obiekt HTTP – dane różnego rodzaju zażądane przez klienta, a wysłane przez serwer w odpowiedzi (oryginalna nazwa ang. Entity Body).

Proces pozyskiwania obiektu HTTP przez klienta można opisać w następujący sposób. Na wstępie, jeśli klient nie łączył się jeszcze z danym serwisem webowym, proces nazywany resolver'em DNS łączy się z serwerem pośredniczącym DNS, który w imieniu klienta przekształca nazwę mnemoniczną serwisu webowego (np. www.po.opole.pl) w adres IP. Następnie po otrzymaniu adresu IP klient nawiązuje połączenie TCP z serwisem webowym występującym pod wskazanym adresem i wysyła do serwisu żądanie HTTP. W kolejnym kroku serwis webowy przesyła do klienta odpowiedź HTTP wraz z żądanym obiektem HTTP. W ramach tego samego połączenia TCP klient może wysłać i odebrać wiele żądań i odpowiedzi HTTP.

Przeglądarki internetowe służą do wyświetlania stron internetowych na komputerach użytkowników. Przeglądarka przed wyświetleniem strony pobiera obiekt będący szkieletem strony HTLM, interpretuje go, a następnie pobiera pozostałe obiekty zagnieżdżone. Źródłem stron HTML są serwisy webowe.

Klient jest źródłem żądań HTTP. Klientem jest komputer wraz z oprogramowaniem, które wysyła żądania HTTP w celu pobrania z serwisu webowego obiekty HTTP [103]. W skład oprogramowania klienta może wchodzić przeglądarka internetowa lub inne oprogramowanie przeszukujące treści w serwisach webowych.

Użytkownikiem jest osoba korzystająca z programu przeglądarki internetowej.

Serwis webowy składa się z serwerów WWW, innych serwerów stanowiących źródło danych, takich jak serwery aplikacji i bazodanowe oraz urządzeń sieciowych niezbędnych do funkcjonowania serwisu. Serwis webowy musi posiadać co najmniej jeden serwer WWW.

Serwis webowy udostępnia użytkownikom żądane treści. Przyjmuje się, że treści pobierane przez użytkowników, czyli strony WWW umieszczone na dyskach komputerów serwisu lub generowane na bieżąco w momencie nadejścia żądania HTTP, stanowią część serwisu webowego. Serwis webowy, składa się więc nie tylko ze sprzętu oraz oprogramowaniem, w jego skład wchodzą również oferowane treści.

Serwer WWW jest pierwotnym źródłem odpowiedzi HTTP. Serwer WWW jest to oprogramowanie udostępniające obiekty HTTP. Najpopularniejszymi serwerami WWW są: Apache [8] oraz serwer WWW wchodzący w skład pakietu Internet Information Services [93] firmy Microsoft Corporation. W dalszej części pracy określenie serwer WWW oznaczać będzie nie tylko samo

oprogramowanie, lecz również komputer, na którym oprogramowanie serwera pracuje.

System webowy, jak zostało to już opisane wcześniej, jest to system składający się z serwisu webowego oraz odpowiednich urządzeń wraz z oprogramowaniem wspierającym obsługę oraz dostarczanie żądań od klienta i odpowiedzi HTTP z serwisu. Elementy systemu webowego nie będące częścią serwisu webowego muszą być zarządzane przez administratora serwisu lub administrator musi mieć w pewnym stopniu nad nimi kontrolę. Do takich elementów zaliczyć na przykład można: elementy systemu DNS wspierające dystrybucję żądań HTTP w systemie, serwery pośredniczące przekierowujące żądania do klastrów serwerów webowych globalnie rozmieszczonych, serwery szeregujące żądania na wejściu do serwisów webowych, system dostarczania treści CDN, którego usługi zostały wykupione dla wsparcia wybranych usług serwisu webowego oraz inne.

1.2. KLASYFIKACJA SYSTEMÓW WEBOWYCH Z JAKOŚCIĄ USŁUG

Internet jest specyficznym medium dającym użytkownikowi bardzo dużą swobodę w wyborze źródła informacji i zmiany tego źródła. Oznacza to, że użytkownicy, których wymagania w danym serwisie webowym nie zostaną spełnione w oczekiwanym zakresie, mogą w łatwy i szybki sposób zmienić źródło informacji i przenieść się do innego serwisu, bez ponoszenia konsekwencji finansowych. Właściciele serwisów webowych, którym zależy na przyciągnięciu klientów, muszą prezentować atrakcyjne treści, równocześnie zapewniając odpowiednią jakość obsługi z technicznego punktu widzenia. O ile użytkownicy nie zwrócą wagi na techniczną stronę obsługi, jeśli będzie prowadzona na wysokim poziomie, o tyle niski poziom obsługi może spowodować rezygnację z korzystania z usług danego serwisu.

Zgodnie ze starszym podejściem, stosowanym jeszcze kilka lat temu, na jakość usług serwisu webowego mają wpływ [53, 126]:

- dostępność – określa czas, w którym serwis prawidłowo realizuje swoje zadania w stosunku do całego czasu obserwacji,
- wydajność – jest różnie definiowana w zależności od potrzeb. Najczęściej miarami wydajności są: liczba obsłużonych żądań na sekundę, przepustowość, czyli liczba danych przesłanych w jednostce czasu, liczba nieprawidłowo obsłużonych żądań w jednostce czasu i inne,
- czas dostępu do usługi – określa, jak długo użytkownicy muszą oczekiwać na realizacją powierzonego zadania,
- bezpieczeństwo – dotyczy bardzo wielu aspektów realizacji usługi takich jak np.: poufność danych, pewność co do prawidłowej realizacji powierzonego zadania, integralność danych i wiele innych,

- osiągalność, która może określać np. lokalizacje, w ramach których dana usługa jest dostępna lub narzędzia techniczne umożliwiające osiągnięcie dostępu.

Obecnie jakość działania systemów webowych jest często oceniana z trzech różnych punktów widzenia. Duża część serwisów internetowych oparta jest na współpracy firm dostarczających usługi internetowe w postaci sprzętu komputerowego i infrastruktury umożliwiającej dostarczenie stron internetowych do użytkowników (jest to tzw. hosting) oraz firm wykupujących usługi hostingu w celu prezentacji w Internecie opracowanych treści. Dlatego też można tu wyróżnić trzy strony zaangażowane w proces obsługi danego serwisu webowego: dostawcę usług internetowych realizującego hosting, właściciela treści korzystającego z hostingu oraz użytkownika korzystającego z oferowanych treści.

Z ekonomicznego punktu widzenia każda z tych stron będzie próbowała osiągnąć jak największe dla siebie korzyści. Dlatego też dostawca usług internetowych będzie dążył do skonstruowania takiej platformy sprzętowo-programowej, aby przy jak najmniejszych kosztach osiągnąć jak największą wydajność systemu tak, by móc oferować usługi dużej liczbie właścicieli treści webowych. Z punktu widzenia dostawcy usług, jakość proponowanej przez niego platformy sprzętowo-programowej będzie oceniana na podstawie wydajności oraz dostępności.

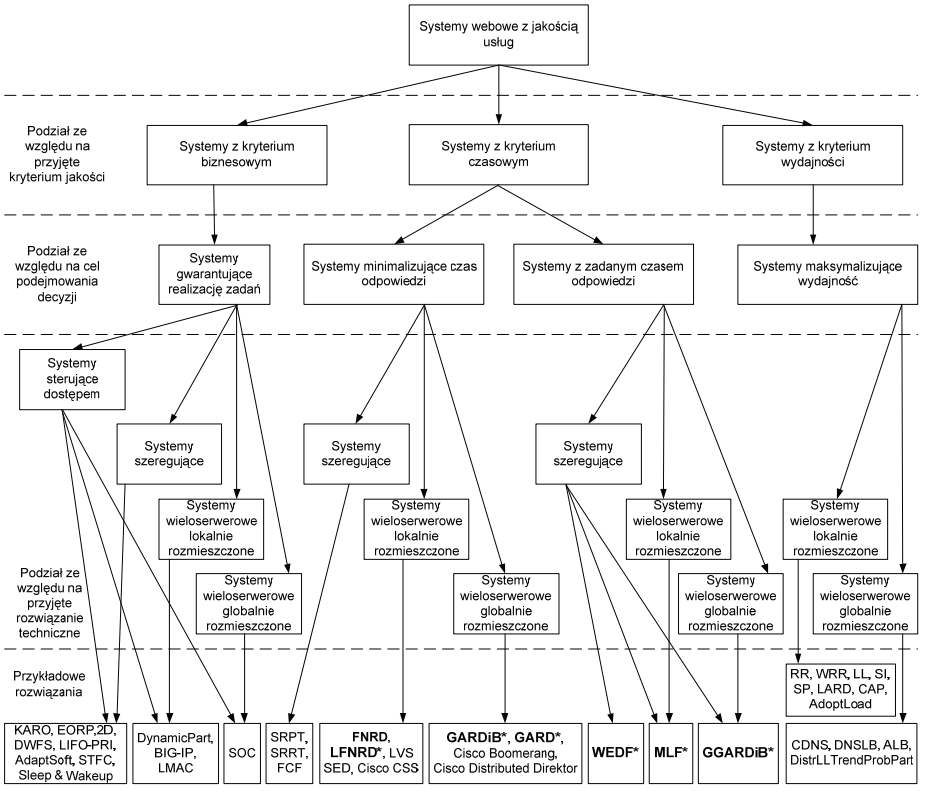
Użytkownicy serwisu, poza jakością dostarczanych treści, będą oceniali najczęściej czas dostarczenia treści oraz dostępność serwisu. Użytkowników nie będzie interesowała wydajność serwisu i techniczne aspekty jego realizacji. Użytkownicy oceniać będą serwis, obserwując pracę swojej przeglądarki internetowej maskującej techniczne aspekty działania serwisu.

Firmę, korzystającą z usługi hostingu będzie interesowało, po zainwestowaniu pieniędzy w przygotowanie godnych uwagi treści, osiągnięcie celów biznesowych, czyli najczęściej uzyskanie jak największego zysku. Firmie takiej będzie zależało, aby stworzony przez nią serwis był dostępny oraz aby rozwiązania techniczne wspierały realizację celów biznesowych nie tylko poprzez prawidłową obsługę żądań HTTP.

Ze względu na wskazane kryteria oceny i potrzeby różnych grup zaangażowanych w działanie i wykorzystanie serwisów internetowych systemy webowe są projektowane tak, aby spełniały postawione cele. Na rys. 1.2 zaprezentowana została klasyfikacja systemów webowych wykonana dla celów tego opracowania.

W klasyfikacji wzięte zostały pod uwagę ważniejsze, uznane rozwiązania oraz rozwiązania autora monografii, w których każde źródło odpowiedzi HTTP jest źródłem pierwotnym (żądania HTTP obsługiwane są w serwerach WWW) [211]. Dlatego też nie są rozważane zarówno w klasyfikacji jak i w dalszej części pracy systemy CDN, wykorzystujące w swej konstrukcji serwery typu casching proxy, obsługujące żądania HTTP i udostępniające obiekty HTTP pobrane

wcześniej z serwerów WWW [32, 38, 136, 169]. Nie są również rozważane systemy związane z obsługą strumieni danych oraz sesji SSL.



Rys. 1.2. Klasyfikacja systemów webowych z jakością usług
(gwiazdką oznaczone zostały systemy prezentowane w pracy)

W klasyfikacji wyszczególnione zostały cztery poziomy. Na poziomie pierwszym, najwyższym dokonany został podział systemów webowych ze względu na podstawowe kryteria brane pod uwagę w procesie opracowywania sposobu pracy danego systemu. Kryteriami tymi są:

- wydajność systemu webowego – systemy z tym kryterium działają w taki sposób, aby podnosić wydajność systemu lub nie dopuszczać do przeciążenia systemu lub jego elementów,
- czas realizacji zadań – systemy z tym kryterium działają tak, aby istniała możliwość zarządzania czasem odpowiedzi ocenianym z punktu widzenia użytkownika, przy czym pojęcie „czas odpowiedzi” może być różnie definiowane w zależności od potrzeb,
- kryterium biznesowe – mówiąc o systemach z kryterium biznesowym mamy na myśli systemy, w których zawarte zostały odpowiednie rozwiązania

techniczne, wpływające bezpośrednio na dochodowość przedsięwzięcia internetowego.

Przyjmuje się, że w systemach z kryterium wydajności i kryterium czasowym wszyscy klienci traktowani są w jednakowy sposób niezależnie od korzyści, jakie mogą przynieść właścicielom serwisu webowego. Różnicowanie klientów pod względem jakości usług realizowane jest jedynie w systemach z kryterium biznesowym.

Prezentowany tu podział nie jest całkowicie jednoznaczny, ponieważ dla przykładu zastosowanie systemów z kryterium wydajności może wpłynąć pozytywnie na dochodowość przedsięwzięcia, jednakże wykorzystanie odpowiedniej metody i algorytmu podejmowania decyzji nie będzie wpływać bezpośrednio na dochodowość serwisu, lecz na przykład na ogólne podniesienie jakości usług całego systemu.

Na poziomie drugim podział systemów dokonany został ze względu na cele podejmowanych decyzji przy przyjętych wcześniej kryteriach. Dla systemów z kryterium biznesowym wyróżnione zostały systemy gwarantujące realizację zadań. Systemy z kryterium czasowym podzielone zostały na systemy minimalizujące czas odpowiedzi oraz systemy z zadanym czasem odpowiedzi. Systemy z kryterium wydajności podzielić można na systemy maksymalizujące wydajność oraz systemy nie dopuszczające do przeciążeń całego systemu lub pewnych jego elementów (na rys. 1.1 zaznaczone zostały jedynie systemy maksymalizujące wydajność ze względu na stosowanie w obu przypadkach tych samych środków umożliwiających osiągnięcie celu).

Na poziomie trzecim wskazane zostały systemy grupowane na podstawie rozwiązań technicznych umożliwiających osiąganie postawionych celów. Grupy te mogą się powtarzać dla systemów z różnymi celami podejmowania decyzji. Wśród grup systemów wyróżnione zostały:

- systemy szeregujące żądania – systemy, w których planowana jest w odpowiedni sposób kolejność obsługi żądań zgodnie z przyjętą polityką inną niż FIFO,
- systemy sterujące dostępem (inaczej z kontrolą przyjęć) – systemy, w których wykorzystuje się sterowanie dostępem (ang. admission control), czyli odrzucanie żądań HTTP, zgodnie z przyjętą procedurą klasyfikacji żądań,
- systemy wieloserwerowe lokalnie rozmieszczone – opisane już wcześniej systemy, które w swej konstrukcji zawierają serwery WWW połączone ze sobą siecią lokalną i znajdujące się w jednej lokalizacji (pomieszczeniu lub budynku). W praktyce systemy wieloserwerowe lokalnie rozmieszczone są najczęściej klastrami serwerów WWW i w dalszej części tego opracowania, pisząc o wieloserwerowych systemach lokalnie rozmieszczonych, będziemy mieli na myśli klastry serwerów webowych,
- systemy wieloserwerowe globalnie rozmieszczone – opisane już wcześniej systemy, w których poszczególne serwery WWW lub nawet całe klastry serwerów zostały umieszczone w różnych lokalizacjach geograficznych.

Systemy webowe opisywane w publikacjach lub też te wdrożone, często są hybrydami systemów wyróżnionych ze względu na stosowane rozwiązania techniczne.

Systemy sterujące dostępem spotykane są jedynie w grupie systemów z kryterium biznesowym ze względu na to, że różnicowanie klientów i odrzucanie ich żądań HTTP ma na celu podnoszenie przychodu osiąganego w ramach serwisu webowego.

Na czwartym poziomie wskazane zostały najważniejsze, wybrane przykładowe rozwiązania spotykane w systemach produkcyjnych lub dobrze przebadane rozwiązania opisane w opracowaniach naukowych.

Na początek opisana zostanie grupa rozwiązań najbardziej znanych i często spotykanych w rozwiązaniach praktycznych, a związana z systemami z kryterium wydajności. Praktycznie rzecz ujmując, prawie wszystkie systemy z kryterium wydajności są konstruowane tak, aby podnosić wydajność systemu. Podniesienie wydajności, w najprostszym przypadku, można osiągnąć poprzez zastosowanie wydajniejszego sprzętu komputerowego lub oprogramowania (rozwiązanie to nie zostało wskazane na rys. 1.1). Metoda ta jednak nie umożliwia skalowania systemu. Systemy webowe, zawierające w swej konstrukcji wiele serwerów webowych, umożliwiają skalowanie systemu, a osiąganie z ich zastosowaniem wysokiej wydajności związane jest z odpowiednią dystrybucją żądań HTTP pomiędzy serwery WWW. Taka dystrybucja obciążeń jest często nazywana równoważeniem obciążeń (ang. Load Balancing) lub też dzieleniem obciążeń (ang. Load Sharing). Problem równoważenia obciążeń był rozpatrywany w informatyce od momentu, gdy pojawiły się pierwsze systemy zawierające wiele zasobów (realizatorów) tego samego typu. Techniki równoważenia obciążeń zostały przeniesione na platformę systemów webowych w momencie, gdy WWW zyskało dużą popularność. Dlatego też dość często, mówiąc o dystrybucji żądań, mówi się o równoważeniu obciążeń. W tym opracowaniu podobnie jak w [53] równoważenie obciążeń oznacza dystrybucję żądań w wieloserwerowych systemach webowych, prowadzącą do wyrównania obciążeń na wszystkich serwerach WWW. Natomiast dzielenie obciążeń oznacza taką dystrybucję żądań, która nie dopuszcza do przeciążenia żadnego z elementów systemu webowego.

Najczęściej wykorzystywanymi w praktycznych rozwiązaniach algorytmami równoważenia obciążeń w systemach lokalnie rozmieszczonych są: algorytm karuzelowy (ang. Round-Robin) oznaczony jako RR, ważony karuzelowy (ang. Weighted Round-Robin) oznaczony jako WRR oraz algorytm wybierający serwer najmniej obciążony LL (ang. Least Loaded) [81]. Algorytmy WRR oraz LL w procesie podejmowania decyzji biorą pod uwagę obciążenie, które może być wyrażone z wykorzystaniem różnych miar np. obciążenia procesora i dysku, liczby równocześnie obsługiwanych żądań, zajętości pamięci. Jednym z ciekawszych algorytmów równoważenia obciążeń jest algorytm Client-Aware Policy [56], w skrócie CAP, w którym wyróżnia się różne klasy żądań, a następnie w ramach każdej z klas z osobna dystrybuuje się żądania zgodnie

z algorytmem RR. Innym algorytmem umożliwiającym dzielenie obciążenia jest algorytm Locality-Aware Request Distribution, w skrócie LARD, który bierze pod uwagę zawartość pamięci podręcznych serwerów WWW [12]. Natomiast w [148] opisany został jeden z pierwszych algorytmów adaptacyjnych: AdaptLoad. Zgodnie z tym algorytmem poszczególne serwery WWW obsługiwały żądania dotyczące obiektów HTTP o określonej wielkości, przy czym wielkość obsługiwanych obiektów zmieniała się dynamicznie w zależności od obciążenia serwerów. Algorytm ten należy do grupy algorytmów, dla której stosuje się wspólną nazwę Service Partitioning (SP) [53]. W algorytmach tej grupy żądania klasyfikowane są na podstawie określonych przesłanek (nazwy żądanego obiektu, wielkości obiektu itp.), a następnie przekierowywane są do serwerów odpowiedzialnych za obsługę żądań danego rodzaju. Innym często stosowanym w praktycznych rozwiązaniach algorytmem jest Session Identifiers (SI), który przydziela użytkowników zawsze do tych samych serwerów, do których użytkownicy przydzieleni zostali przy wysłaniu pierwszego żądania HTTP [75].

W rozwiązaniach umożliwiających podnoszenie wydajności w systemach wieloserwerowych globalnie rozmieszczonych, często w dystrybucji żądań wykorzystuje się system DNS, umożliwiający przekierowywanie żądań klientów do poszczególnych serwerów webowych lub klastrów serwerów jak np. w rozwiązaniu opisanym w [51] o nazwie Cluster DNS (CDNS) lub urządzeniach Network Dispatcher [94] oraz Distributed Director [65]. Dodatkowo możliwe jest również dystrybuowanie obciążenia w systemach webowych rozproszonych globalnie, wykorzystując mechanizm przekierowywania żądań zastosowany w protokole HTTP, a polegający na przesłaniu do klienta informacji, że żądany obiekt znajduje się w innej lokalizacji niż wskazuje na to adres zawarty w żądaniu. Przykładowymi rozwiązaniami wykorzystującymi ten sposób jest system prezentowany w [7] lub przełącznik webowy AppDirector firmy Radware [144, 145].

Systemy webowe globalnie rozmieszczone wykorzystują najczęściej takie algorytmy dystrybucji żądań jak [68]: RR, WRR (w którym wagi są stałe lub są ustalane dynamicznie na podstawie obciążenia poszczególnych elementów systemu), algorytmy podejmujące decyzję o przekierowaniu żądań na podstawie adresu IP klienta. W nowszych opracowaniach naukowych proponuje się stosowanie algorytmów dynamicznych oraz adaptacyjnych w dystrybucji żądań. Przykładem takiego rozwiązania jest system DNS-Based Load Balancing (DNSLB) [89], dla którego proponowana jest odpowiednia architektura całego systemu oraz dostosowany do niej adaptacyjny algorytm dystrybucji. W innych rozwiązaniach proponuje się adaptacyjne algorytmy dystrybucji wykorzystujące informacje o obciążeniu serwerów WWW jak w przypadku systemów CDNS [51] oraz Authoritative Domain Name Server (ADNS) [69], które w zależności od stanu systemu webowego zmieniają przedział czasowy, przez który można przechowywać informacje przekazane przez serwer DNS. Jedną z najnowszych opisanych metod dystrybucji żądań w systemach globalnie rozproszonych jest DistrLLTrendProbPart [7], który adaptacyjnie dostosowuje się do zmian zacho-

wania klientów tak, aby nie dopuszczać do przeciążeń lokalnych klastrów serwerów webowych. W metodzie tej do dystrybucji żądań wykorzystywane są zarówno mechanizmy systemu DNS oraz mechanizm przekierowywania żądań zastosowany w protokole HTTP.

Zostaną teraz opisane przykładowe rozwiązania stosowane w systemach webowych z kryterium biznesowym. Systemy tego typu nazywane są bardzo często systemami e-commerce. Dla systemów z kryterium biznesowym tworzy się nowe wskaźniki jakości obrazujące, w jaki sposób wykorzystana technologia wspiera zarabianie pieniędzy. Najczęściej w systemach e-commerce do utrzymania jakości usług wykorzystuje się metody związane ze sterowaniem dostępem oraz szeregowaniem żądań HTTP. Systemy e-commerce często gwarantują realizację usług użytkownikom, którzy rozpoczęli pracę z danym serwisem tak, aby mogli zakończyć pracę i sfinalizować, kluczowe z punktu widzenia właścicieli serwisu, działania, dla przykładu dokończyć zakupy z sklepie internetowym. Przykładami takich rozwiązań są systemy: Dynamic Weighted Fair Sharing (DWFS) [58], LIFO-Priority (LIFO-Pri) [160], Session-based two-dimensional service (2D) [228], AdaptSoft [1], Simulated Annealing Load Spreading Algorithm (SALSA) [23].

Inne podejście polega na dobrej obsłudze tych klientów serwisu, którzy należą do grupy klientów uprzywilejowanych i mogą przynieść potencjalne zyski lub którzy zapłacili za lepszą obsługę. W rozwiązaniach realizujących to podejście często stosuje się sterowanie dostępem oraz złożone algorytmy szeregowania żądań HTTP. Przykładowymi rozwiązaniami tego typu są: E-commerce Oriented Resource Policy (EORP) [125], Key customers And Revenue Oriented admission and scheduling (KARO) [40], Self-tuning Fuzzy Contoroller (STFC) [173], Sleep and Wakeup [3]. Innym stosowanym również od wielu lat rozwiązaniem jest wykorzystanie klastrów serwerów webowych, w których przełączniki webowe, zarządzające przepływem żądań HTTP, przekierowują żądania uprzywilejowanych klientów na wydzielone, nieobciążone serwery. Przykładowymi rozwiązaniami tego typu są: DynamicPart [6], BIG-IP [75], Latency Targeted Multi-Class Admission Control (LMAC) [100]. Wśród licznych publikacji dotyczących systemów webowych z jakością usług nie ma wielu propozycji systemów wieloserwerowych globalnie rozmieszczonych należących do grupy systemów z kryterium biznesowym. Do nielicznych rozwiązań tego typu wykorzystujących sterowanie dostępem należy system Self Overload Control (SOC) [17].

Zostaną teraz omówione przykładowe systemy z kryterium czasowym. Wśród prezentowanych rozwiązań znajdują się systemy opracowane przez autora pracy (oznaczone czcionką pogrubioną na rys. 1.1) oraz systemy prezentowane w dalszej części monografii (oznaczone gwiazdką na rys. 1.1).

Projektowanie systemów webowych, w którym brany jest pod uwagę czas odpowiedzi wymaga zazwyczaj opracowania odpowiedniego modelu systemu webowego umożliwiającego szacowanie czasów odpowiedzi i podejmowania decyzji w czasie rzeczywistym. W dużej części rozwiązań czasy odpowiedzi

szacowane są na podstawie średnich czasów odpowiedzi uzyskanych po obsłużeniu większej liczby żądań jak np. w [87, 154]. Inny model systemu obsługi żądań w serwisie zaproponowany został w [25, 33, 195]. Model ten zawiera w swej konstrukcji model rozmyto-neuronowy.

Jak zostało już wcześniej wspomniane, wyróżnione zostały systemy z kryterium czasowym minimalizujące czas odpowiedzi oraz systemy z zadanym czasem odpowiedzi. Wśród badanych systemów minimalizujących czasy odpowiedzi na żądania HTTP istnieje kilka rozwiązań wykorzystujących szeregowanie żądań. W rozwiązaniach tych często wykorzystywana jest polityka Shortest-Remaining-Processing-Time (SRPT), zgodnie z którą w pierwszej kolejności obsługiwane są żądania, których obsługa może być najszybciej zakończona. Zastosowanie opisywanego sposobu umożliwia skrócenie średniego czasu odpowiedzi dla żądań. Stosowanie polityki SRPT może jednak powodować wydłużenie czasu odpowiedzi dla żądań, dla których czas ten jest już i tak długi [4, 19]. W konsekwencji może to prowadzić do niezadowolenia użytkowników i przerywania korzystania z serwisu w ważnych momentach np. w czasie składania zamówień w sklepie internetowym. Wśród znanych rozwiązań stosujących szeregowanie z wykorzystaniem omawianej polityki i pewnych jej odmian wymienić można: Shortest Remaining Processing Time SRPT [87, 154], Fastest Connection First (FCF) [128], Shortest Remaining Response Time (SRRT) [4].

Zostaną teraz przedstawione systemy umożliwiające minimalizację czasu odpowiedzi w lokalnie rozmieszczonych systemach wieloserwerowych, a w szczególności w klastrowych systemach webowych. Wśród niewielu opracowań tego typu systemów wyróżnić można systemy Fuzzy Neural Request Distribution (FNRD) [26, 28, 29] wykorzystujący modele rozmyto-neuronowy oraz kolejną wersję tego systemu: Local Fuzzy Neural Request Distribution (LFNRD) [218, 220] wykorzystując bardziej zaawansowany model rozmyto-neuronowy zaproponowany w [34]. Opis działania systemu LFNRD zaprezentowany jest w rozdziale 3.1. Znane są również rozwiązania stosowane w przemysłowych przełącznikach webowych, które w procesie podejmowania decyzji wykorzystują ogólną informację o opóźnieniach w obsłudze żądań, są to np. algorytm Shortest Expected Delay (SED) stosowany w przełączniku programowym Linux Virtual Server (LVS) [109] lub rozwiązanie zaproponowane w serii przełączników Cisco CSS 11500 [65].

Budowa dużego serwisu webowego umożliwiającego minimalizację czasów odpowiedzi wymaga zastosowania systemu wieloserwerowego globalnie rozmieszczonego. W większości rozwiązań należących do tej grupy systemów stosowane są odpowiednio zmodyfikowane systemy DNS umożliwiające przekierowanie żądań klientów do źródeł spełniających odpowiednie kryteria. Stosowane są najczęściej dwa kryteria, są to: odległość geograficzna lub sieciowa danego klienta od źródła (wykorzystywana np. w urządzeniu Cisco Distributed Direktor [64]) lub czas przesłania pojedynczego pakietu IP między klientem a danym źródłem (rozwiązanie wykorzystywane w systemie Cisco Boomerang [66]). Niewiele zostało opracowanych systemów webowych globalnie rozpro-

szonych, działających w taki sposób, aby minimalizować czas odpowiedzi na żądania klientów biorąc pod uwagę równocześnie czas przesyłania danych między klientem a źródłem danych oraz obciążenia źródeł danych. Do nielicznych propozycji tego typu systemów należy system Global Adaptive Request Distribution with Broker (GARDiB) [33, 44, 188] wykorzystujący serwery pośredniczące oraz system Global Adaptive Request Distribution (GARD) [206, 213]. Oba systemy opisane są w rozdz. 3.

Przedstawione zostaną teraz rozwiązania dotyczące systemów z zadanym czasem odpowiedzi. Nie ma wielu systemów opisanych w literaturze, stosujących jedynie odpowiednie metody szeregowania w celu zagwarantowania jakości usług na żądanym poziomie. Wśród nielicznych wyróżnić można system Web Earliest Deadline First (WEDF) [207, 209] umożliwiający obsługę całych stron webowych w żądanym czasie (system opisany jest w rozdz. 4.1). Wśród systemów wykorzystujących lokalne klastry serwerów webowych wyróżnić można system Most Loaded First (MLF) wykorzystujący odpowiedni algorytm szeregowania żądań oraz nowy algorytm dystrybucji, które umożliwiają obsługę całych stron webowych w żądanym czasie. Opis systemu MLF znajduje się w rozdziale 4.2. Innym systemem, również gwarantującym czas odpowiedzi dla całych stron webowych, jest system Guaranteed Global Adaptive Request Distribution with Broker (GGARDiB) [219] (opisany w rozdziale 4.3) stosowany w serwisach webowych zawierających globalnie rozmieszczone klastry serwerów webowych. W systemie tym wykorzystywane są odpowiednie metody szeregowania oraz dystrybucji żądań.

Reasumując, wskazane zostały trzy grupy systemów webowych, w których wykorzystuje się odpowiednie techniki wpływające na podwyższenie lub gwarantowanie jakości usług. Pierwsza grupa systemów to systemy z kryterium wydajności. Druga grupa to systemy, w których rozwiązania techniczne wspierają osiąganie celów biznesowych. Ostatnia grupa wyróżnionych systemów webowych to systemy z kryterium czasowym, które realizują usługi w taki sposób, aby dostarczyć klientom żądane treści w odpowiednim czasie.

W dalszej części monografii przedstawione zostaną projekty systemów webowych z jakością usług uwzględniające kryterium czasowe.

1.3. SYSTEMY Z KRYTERIUM CZASOWYM PREZENTOWANE W MONOGRAFII

Rozwój systemów webowych z jakością usług rozpoczął się od tworzenia systemów z kryterium wydajności, wypełniając równocześnie zapotrzebowanie grupy firm dostarczających usługi internetowe i mających bezpośredni wpływ na budowę i sposób działania systemów webowych. W trakcie bardzo szybkiego rozwoju Internetu dopiero po upływie pewnego czasu dostrzeżono potrzeby klientów końcowych, decydujących o rentowności przedsięwzięć internetowych, w wyniku czego rozpoczęto tworzenie rozwiązań dla systemów z kryterium cza-

sowym. Jednym z pierwszych systemów webowych uwzględniających czas odpowiedzi na żądania HTTP był system Fuzzy Adaptive Request Distribution (FARD) [25, 189, 191, 192], wczesna wersja systemu FNRD* i prezentowanego w pracy LFNRD.

Projektowanie systemów webowych z kryterium czasowym nie jest proste ze względu na trudności w opracowywaniu efektywnych czasowo algorytmów podejmowania decyzji.

W niniejszej monografii prezentowany jest kompleks rozwiązań stosowanych w systemach webowych uwzględniających kryterium czasowe. Proponowane rozwiązania były rozwijane od momentu powstania grupy systemów webowych z kryterium czasowym i stanowią prekursorskie rozwiązania w tym zakresie. W ramach monografii prezentowane są systemy minimalizujące czasy odpowiedzi, jak również systemy z zadanym czasem odpowiedzi. Opracowane rozwiązania mogą być stosowane w systemach webowych różnej skali, począwszy od systemów zawierających jeden serwer webowy, poprzez systemy kilkuserwerowe, kończąc na systemach zawierających kilkadziesiąt serwerów umieszczonych w różnych lokalizacjach.

Prezentowane w ramach pracy podejścia są uznane, jak również stanowiły inspirację w tworzeniu kolejnych systemów, między innymi [27, 35, 40, 43, 80 – 84, 149, 170, 181, 227].

W monografii przedstawione są projekty sześciu systemów webowych z kryterium czasowym. Dla grupy systemów minimalizujących czasy odpowiedzi dla poszczególnych żądań HTTP opracowane zostały wspomniane już wcześniej systemy LFNRD, GARDiB* oraz GARD*. System LFNRD wykorzystuje w swej konstrukcji klaster serwerów webowych lokalnie rozmieszczonych. Systemy GARDiB oraz GARD przeznaczone są do stosowania w serwisach webowych zawierających kilka klastrów serwerów webowych rozmieszczonych w różnych lokalizacjach geograficznych.

Dla grupy systemów z zadanym czasem odpowiedzi dla stron internetowych, opracowane zostały trzy systemy: WEDF*, MLF i GGARDiB. W systemach tych wykorzystane zostały techniki szeregowania żądań bez sterowania dostępem, co stanowi nowość. Dzięki takiemu podejściu jakość usług zapewniana jest dla wszystkich użytkowników, nie tylko dla wybranych grup użytkowników, którzy potencjalnie mogą przynieść zyski.

System WEDF przeznaczony jest do stosowania w serwisach zawierających jeden serwer WWW. System MLF przeznaczony jest do zastosowania w serwisach zawierających klaster serwerów webowych umieszczonych w jednej lokalizacji. Natomiast system GGARDiB podobnie jak GARDiB stosowany jest w serwisach webowych zawierających wiele klastrów serwerów webowych rozmieszczonych w różnych lokalizacjach geograficznych.

* System opracowany w ramach pracy naukowej finansowanej ze środków na naukę w latach 2006–2009 jako projekt badawczy nr N516 032 31/3359.

Dla każdego z opisywanych w monografii systemów przedstawiony jest projekt systemu, w skład którego wchodzi:

- ogólna koncepcja prezentująca budowę i działanie systemu oraz metody w nim wykorzystywanej,
- zadanie projektowe do rozwiązania,
- projekt urządzenia lub urządzeń sterujących pracą systemu zgodnie z przyjętą metodą i algorytmami,
- opis symulacyjnego stanowiska badawczego oraz wyniki badań wraz z dyskusją[†].

Opracowanie wymienionych powyżej systemów webowych wymagało wcześniejszego opracowania modelu systemu obsługi żądań HTTP umożliwiającego szacowanie czasów odpowiedzi na żądania HTTP dla wybranych elementów systemów webowych. Przyjęty model to model rozmyto-neuronowy umożliwiający podejmowanie decyzji w czasie rzeczywistym, jak również w sposób adaptacyjny dostosowujący swoje działanie do działania obiektu. W prawie wszystkich opisanych dalej systemach (oprócz systemu WEDF) wykorzystany został ten sam model. Wykorzystanie wskazanego modelu w poszczególnych systemach tworzy wspólną platformę koncepcyjną dla prezentowanych rozwiązań. Systemy wykorzystujące wskazane metody są adaptacyjnymi systemami podejmowania decyzji, w których wyróżnić można pewne algorytmy podejmowania decyzji nazywane również algorytmami podstawowymi oraz algorytmy adaptacji, które z wykorzystaniem pewnych dodatkowych informacji, korygują działania algorytmów podstawowych [48].

Ponieważ rozmyto-neuronowy model systemu obsługi żądań HTTP stanowi integralną część opisywanych dalej systemów, dlatego też zostanie przedstawiony przed ich opisem.

1.4. CEL ORAZ ZAKRES MONOGRAFII

Celem pracy jest opracowanie grupy systemów webowych z kryterium czasowym, które umożliwiają podnoszenie jakości oferowanych usług poprzez minimalizację czasów odpowiedzi na żądania HTTP oraz utrzymywanie czasów odpowiedzi nie dłuższych niż założone dla całych stron WWW dla małych serwisów webowych zawierających jeden lub kilka serwerów WWW oraz dużych serwisów zawierających do kilkudziesięciu serwerów WWW.

W ramach monografii zostaną przedstawione projekty systemów webowych umożliwiających:

[†] Badania nad systemami webowymi opisanymi w pracy przeprowadzone zostały w Laboratorium Rozproszonych Systemów Komputerowych Zakładu Rozproszonych Systemów Komputerowych Instytutu Informatyki Politechniki Wrocławskiej.

- podnoszenie jakości poprzez minimalizację czasów odpowiedzi na żądania HTTP w systemach wieloserwerowych lokalnie rozmieszczonych oraz systemach wieloserwerowych globalnie rozmieszczonych,
- utrzymanie czasów odpowiedzi nie dłuższych niż założone dla całych stron WWW w systemach webowych zawierających jeden serwer WWW, systemach wieloserwerowych lokalnie rozmieszczonych oraz systemach wieloserwerowych globalnie rozmieszczonych.

Monografia podzielona została na pięć rozdziałów. W rozdziale drugim przedstawiony został projekt rozmyto-neuronowego modelu systemu obsługi żądań HTTP, umożliwiający szacowanie czasów odpowiedzi w systemach webowych.

W rozdziale trzecim zaprezentowane zostały systemy webowe umożliwiające minimalizację czasów odpowiedzi na żądania HTTP. W ramach rozdziału przedstawione zostały projekty systemów webowych LFNRD, GARDiB i GARD.

W rozdziale czwartym rozpatrywane są zagadnienia gwarantowania jakości usług w systemach webowych, prezentowane są projekty systemów WEDF, MLF oraz GGARDiB.

Rozdział piąty stanowi podsumowanie, w ramach którego wskazane zostały wykonane zadania oraz kierunki dalszych prac.

2. MODEL SYSTEMU OBSŁUGI ŻĄDAŃ HTTP

Stosowanie modeli systemów webowych w systemach podejmowania decyzji ma duże znaczenie, ponieważ umożliwia podnoszenie jakości podejmowanych decyzji. Wykorzystanie modeli umożliwia przewidywanie skutków decyzji przed ich ostatecznym podjęciem. W szczególności zastosowanie odpowiedniego modelu w systemach z kryterium czasowym może otworzyć perspektywy w opracowaniu całej klasy nowych metod i rozwiązań.

Model systemu, który mógłby zostać zastosowany w systemach webowych w procesie podejmowania decyzji, posiadać musi odpowiednie cechy:

- praca w czasie rzeczywistym umożliwiająca podejmowanie decyzji w czasie znacznie krótszym od czasu odpowiedzi na żądanie HTTP,
- dopasowywanie się do zmieniających się warunków przetwarzania w sposób adaptacyjny,
- elastyczność w zastosowaniach do prostych oraz złożonych systemów webowych,
- podejmowanie decyzji w warunkach niepewności, gdy informacja o działaniu systemu webowego jest niepełna lub niedokładna,
- łatwość w konfiguracji wstępnej, model nie powinien wymagać złożonej wstępnej konfiguracji przez eksperta dziedzinowego i dopasowania do rzeczywistego systemu przed uruchomieniem systemu.

Proponowany model musi umożliwiać estymowanie czasów odpowiedzi na wysyłane przez klientów żądania HTTP.

Stosowanych jest wiele modeli wspomagających podejmowanie decyzji w warunkach niepewności, są to dla przykładu modele: relacyjne, probabilistyczne, growe i rozmyte [47]. Każdy z wymienionych modeli daje możliwość adaptacyjnego dopasowywania się do zmieniających warunków przetwarzania oraz umożliwia, przy odpowiedniej implementacji, podejmowanie decyzji w czasie rzeczywistym. Wydaje się również, że możliwe byłoby opracowanie odpowiedniego modelu systemu webowego, z wykorzystaniem każdego z wymienionych modeli wspomagających podejmowanie decyzji. Do dalszej analizy przyjęty został model rozmyty, a dokładniej model rozmyto-neuronowy.

Systemy rozmyte i neuronowe są stosowane od dłuższego czasu w licznych, wdrożonych rozwiązaniach praktycznych jak i rozwiązaniach prototypowych. Logika rozmyta została po raz pierwszy wprowadzona przez L.A. Zadeha [182] i od tego czasu była szeroko stosowana do budowy systemów inteligentnych [18, 73, 85, 99, 106, 107, 108, 112, 114, 117, 158]. W 1977 E.H. Mamdani [115] wprowadził procedury wnioskowania rozmytego, co zaowocowało licznymi wdrożeniami nowego rozwiązania. Główną zaletą logiki

rozmytej jest to, że systemy ją stosujące dobrze radzą sobie z informacjami nieprecyzyjnymi często charakteryzującymi systemy fizyczne. Logika rozmyta umożliwia budowę modeli dobrze odpowiadających ludzkiemu sposobowi rozumienia i postrzegania rzeczywistości [18, 116, 138, 163, 178, 183 – 185]. Kompleksowy przegląd z zakresu rozwiązań stosowanych w systemach podejmowania decyzji znaleźć można w [186].

Poprzez zastosowanie w systemach rozmytych rozwiązań z zakresu sieci neuronowych otrzymać można systemy hybrydowe, posiadające dodatkowo możliwości uczenia się [74, 105, 132, 165]. Połączenie sieci neuronowej i logiki rozmytej umożliwia tworzenie kontrolerów, charakteryzujących się możliwościami adaptacji przy równoczesnej zdolności do podejmowania decyzji w środowisku niepewnym i zaszumionym [101, 104, 140, 141, 150, 151, 179]. Systemy rozmyto-neuronowe były już stosowane do sterowania w systemach webowych, na przykład do szeregowania żądań na wejściu do systemu webowego [172, 173], sterowania klastrem serwerów webowych [148] oraz w rozwiązaniach proponowanych przez autora.

W dalszej części rozdziału opisany zostanie model systemu obsługi żądań HTTP. Model ten wykorzystany został w prawie wszystkich rozwiązaniach prezentowanych w monografii. Pierwsza uproszczona wersja modelu rozmyto-neuronowego wykorzystanego do szacowania czasów odpowiedzi na żądania HTTP serwerów webowych została zaprezentowana w [189] oraz dalej w [25, 191, 192]. Zaproponowany model rozmyty umożliwiał jedynie adaptacyjne zmiany parametrów funkcji wyostrzania bez możliwości strojenia pozostałych parametrów modelu. W [26] zaprezentowany został model ze zredukowaną liczbą wejść, dzięki czemu zwiększone zostało tempo adaptacji modelu do zmieniających się warunków przetwarzania. Dalsze prace związane były z poprawą jakości pracy modelu [199] oraz zwiększenia możliwości zastosowań modelu, na przykład jako podstawa do budowy symulatora serwera webowego [195 – 197]. W publikacji [31] zaproponowana została następna wersja modelu umożliwiająca strojenie nie tylko wartości parametrów funkcji wyostrzania modelu rozmytego, ale również parametrów funkcji rozmywania. Zastosowanie nowego modelu prezentowane było między innymi w [32, 33, 36, 80, 207, 210, 213, 219].

2.1. OPIS REALIZOWANEGO ZADANIA

Na wstępie przedstawione zostanie zadanie projektowe, którego rozwiązanie stanowić będzie koncepcję modelu systemu obsługi żądań.

Poniżej zamieszczona została lista oznaczeń potrzebnych do sformułowania zadania opisanego w tym rozdziale. Oznaczenia te będą również wykorzystywane w dalszej części opracowania.

X – jest zbiorem żądań HTTP obsługiwanych prawidłowo w serwisie webowym, w wyniku wykonania których powstaje odpowiedź warto-

ściowa dla użytkownika, $X = \{x_i : x_i$ jest prawidłowym żądaniem HTTP obsługiwanym w systemie, $i = 1,2,...\}$,

x_i – żądanie HTTP, polecenie wykonania operacji pobrania z serwisu webowego obiektu o określonym adresie URI. Przyjmijmy, że $x_i = \langle u_i, ck_i \rangle$ i $x_i \in X$,

i – indeks żądania x_i, $i = 1,2,...$; indeks ten umieszczony przy innych oznaczeniach wskazuje, że brana jest pod uwagę wartość zmiennych w momencie nadejścia żądania x_i, lub zmienne te są związane z konkretnym i-tym żądaniem,

u_i – adres HTTP obiektu żądanego w i-tym żądaniu,

ck_i – informacje zawarte w polu *cookie* i-tego żądania,

$\tilde{t}_i$ – zmierzony czas odpowiedzi i-tego żądania; definicja tego czasu jest przyjmowana w zależności od konkretnego zastosowania. Ogólnie można powiedzieć, że czas ten liczony jest od momentu nadejścia i-tego żądania do wskazanego elementu infrastruktury sieciowej do momentu przekazania odpowiedzi na żądanie do tego samego elementu,

$\hat{t}_i$ – szacowany czas odpowiedzi na i-te żądanie,

O_i – obciążenie systemu obsługi żądań HTTP; we wszystkich omawianych dalej metodach obciążenie systemu będzie reprezentowane przez dwie składowe $O_i = [a_i, b_i]$ wskazujące różne miary obciążenia systemu w zależności od omawianego przypadku,

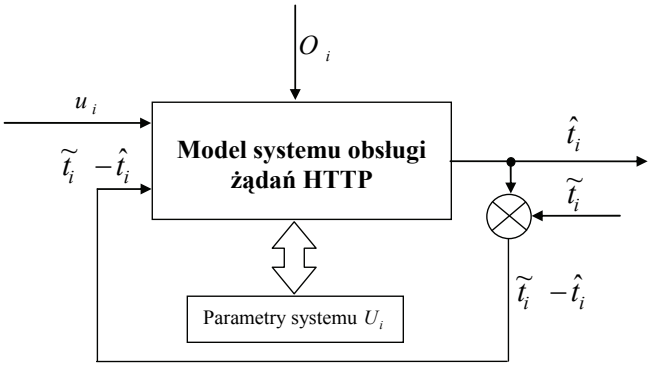
U_i – macierz parametrów modelu systemu obsługi żądań HTTP, $U_i = [U_{1i},...,U_{ki},...,U_{Ki}]$, gdzie $k \in \{1,...,K\}$,

k_i – klasa obiektu żądanego w i-tym żądaniu, $k \in \{1,...,K\}$,

K – liczba wyróżnionych klas obiektów.

Zadanie projektowe: należy opracować model systemu obsługi żądań HTTP o parametrach U_i, który dla żądanego w x_i obiektu o adresie u_i oraz obciążeniu O_i systemu obsługi żądań wyznaczy szacowany czas odpowiedzi $\hat{t}_i$ tak, że $\hat{t}_i \approx \tilde{t}_i$. Dodatkowo model powinien adaptacyjnie dopasowywać się do zmiennych warunków przetwarzania, modyfikując parametry U_i na podstawie wiedzy o zmierzonych czasach odpowiedzi $\tilde{t}_1,...,\tilde{t}_{i-1}$.

Na rys. 2.1 przedstawiony został ogólny schemat systemu obsługi żądań HTTP realizowanego zgodnie przedstawionymi założeniami projektowymi.

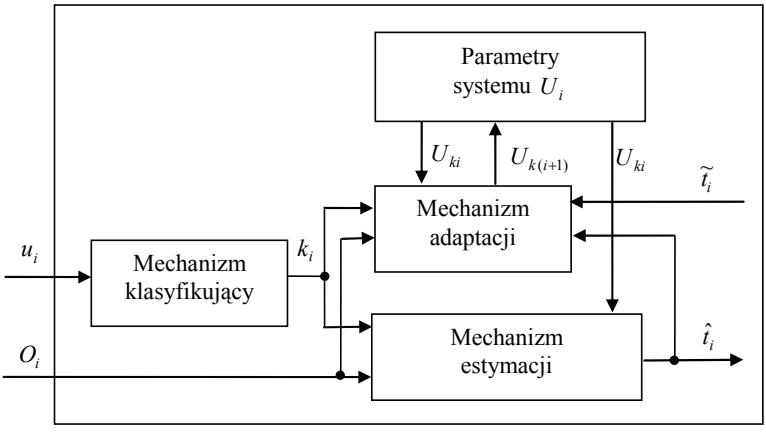


Rys. 2.1. Ogólny schemat modelu systemu obsługi żądań HTTP

Przed rozpoczęciem szczegółowego omawiania sposobu działania systemu obsługi żądań należy jeszcze dokładniej sprecyzować, czym jest żądanie i odpowiedź HTTP. Pojęcie żądania i odpowiedzi HTTP będzie miało w dalszej części tego opracowania podwójne znaczenie. Z jednej strony zarówno żądanie jak i odpowiedź będą informacjami wykorzystywanymi w procesie podejmowania decyzji, z drugiej jednak strony będą to fizyczne nośniki informacji w protokole HTTP przesyłane przez sieć. Dlatego też przepływy dotyczące żądania i odpowiedzi, traktowanych jako informacje, będą na rysunkach oznaczane pojedynczą linią, podobnie jak pozostałe informacje przepływające z jednego podsystemu do drugiego. Przepływ żądania i odpowiedzi traktowanych jako nośniki informacji będą oznaczone podwójną linią.

2.2. OPIS MODELU SYSTEMU OBSŁUGI ŻĄDAŃ HTTP

Model systemu obsługi żądań HTTP składa się z czterech modułów funkcjonalnych: mechanizmu klasyfikującego, mechanizmu estymacji, mechanizmu adaptacji oraz modułu parametrów systemu. Na rys. 2.2 przedstawiony jest schemat modelu systemu obsługi żądań HTTP.



Rys. 2.2. Szczegółowy schemat modelu systemu obsługi żądań HTTP

Moduł mechanizmu klasyfikującego klasyfikuje wszystkie obiekty żądane przez klienta. Na wejściu do modułu przekazywany jest adres u_i żądanego obiektu, gdzie $i = 1,2,....$. Moduł klasyfikujący posiada informacje o wielkości oraz rodzaju obiektów HTTP udostępnianych przez serwis. Obiekty są tak klasyfikowane, aby czasy odpowiedzi na żądania dotyczące obiektów należących do jednej klasy były zbliżone. Obiekty statyczne klasyfikowane są na podstawie ich wielkości. W klasyfikacji powinno brać się pod uwagę czynniki wpływające na czas obsługi żądań statycznych, np. wielkość bloków danych przechowywanych na dyskach twardych lub maksymalną wielkość bloków danych przesyłanych w sieciach komputerowych wykorzystywanych do przekazywania odpowiedzi HTTP.

Obiekty dynamiczne mogą być klasyfikowane w taki sposób, aby każdy obiekt dynamiczny przydzielony był do innej klasy. Sposób klasyfikacji żądanych obiektów może zależeć od sposobów i celu wykorzystania modelu systemu obsługi żądań.

Na wyjściu mechanizmu klasyfikującego przekazywana jest klasa k_i żądanego obiektu, gdzie $k_i = 1,...,K$, a K jest liczbą wyznaczonych klas.

Moduł parametrów systemu przechowuje informacje dotyczące wartości parametrów wykorzystywanych przez mechanizm estymujący do szacowania czasu odpowiedzi na żądanie klienta. Parametry systemu oznaczone zostały $U_i = \left[U_{1i},...,U_{ki},...,U_{Ki}\right]$. Natomiast U_{ki} to parametry dotyczące żądań k-tej klasy i $U_{ki} = \left[A_{ki}, B_{ki}, T_{ki}\right]$, gdzie $A_{ki} = \left[\alpha_{1ki},...,\alpha_{l\,ki},...,\alpha_{(L-1)\,ki}\right]$, $B_{ki} = \left[\beta_{1ki},...,\beta_{m\,ki},...,\beta_{(M-1)\,ki}\right]$, $T_{ki} = \left[t_{1\,ki},...,t_{j\,ki},...,t_{J\,ki}\right]$, $k = 1,...,K$.

Dla uproszczenia zapisu przyjmijmy, że $k = k_i$.

Mechanizm estymacji szacuje czasy odpowiedzi $\hat{t}_i$, $i = 1,2,...$ Czas ten szacowany jest na podstawie: klasy k żądanego obiektu, obciążenia O_i systemu, dla którego budowany jest model oraz aktualnych danych U_{ki}.

Obciążenie O_i systemu może mieć wiele składowych reprezentujących miary obciążenia poszczególnych elementów systemu, zasobów, które mogą stać się potencjalnymi „wąskimi gardłami" lub też, dla których czas przebywania żądań (pobytu w kolejce i obsługi) stanowi znaczącą część czasu odpowiedzi. Właściwym jest takie dobranie składowych obciążenia O_i, aby ich liczba była mała, dzięki czemu proces estymacji przebiegał będzie znacznie szybciej. We wszystkich dalej omawianych algorytmach dystrybucji żądań obciążenie systemu będzie reprezentowane przez dwie składowe a_i oraz b_i, reprezentujące różne miary obciążenia systemu w zależności od omawianego przypadku.

Mechanizm adaptacji aktualizuje informacje U_{ki} dotyczące k-tej klasy obiektów zawarte w module parametrów systemu U_i. Mechanizm adaptacji dokonuje modyfikacji, posiadając informacje o obciążeniu systemu O_i, klasie k żądanego obiektu oraz estymowanym $\hat{t}_i$ i zmierzonym czasie odpowiedzi $\tilde{t}_i$. W wyniku działania mechanizmu adaptacyjnego wyznaczane są uaktualnione parametry systemu $U_{k(i+1)}$.

Model systemu obsługi żądań HTTP może pracować w dwóch trybach: estymacji i adaptacji. W trybie estymacji model systemu szacuje czas odpowiedzi na żądanie. W trybie tym mechanizm adaptacji nie bierze udziału w obliczeniach. Model serwisu webowego pracując w trybie adaptacji, dostosowuje się do środowiska, w którym pracuje, poprawiając jakość pracy w trybie estymacji. Adaptacja realizowana jest przez mechanizm adaptacji po zakończeniu obsługi żądania klienta i uzyskania zmierzonego czasu odpowiedzi $\tilde{t}_i$ na żądanie.

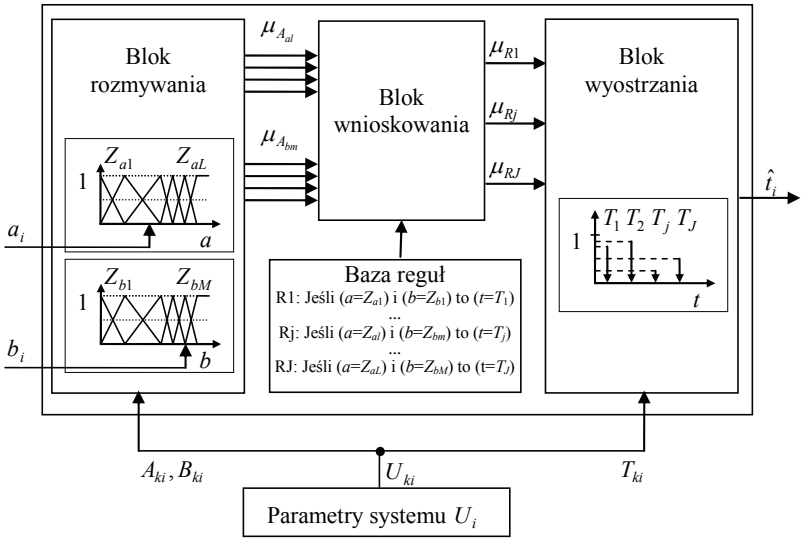
Mechanizm estymacji oraz adaptacji tworzą rozmyto-neuronowy model systemu webowego, którego wagi dla poszczególnych klas obiektów przechowywane są w module parametrów systemu.

2.2.1. MECHANIZM ESTYMACJI

Działanie modelu systemu obsługi żądań w trybie estymacji jest równoważne działaniu modelu rozmytego, którego schemat przedstawiony jest na rys. 2.3. Model składa się z następujących bloków:

- rozmywania (fuzyfikacji), który odpowiedzialny jest za przekształcanie nie rozmytych danych wejściowych (wartości miar obciążeń systemu) w zbiory rozmyte,
- bazy reguł,

- wnioskowania, w którym obliczane są stopnie przynależności do poszczególnych reguł,
- wyostrzania (defuzyfikacji), którego zadaniem jest wyliczenie ostrej wartości wyjścia.



Rys. 2.3. Schemat działania mechanizmu estymującego

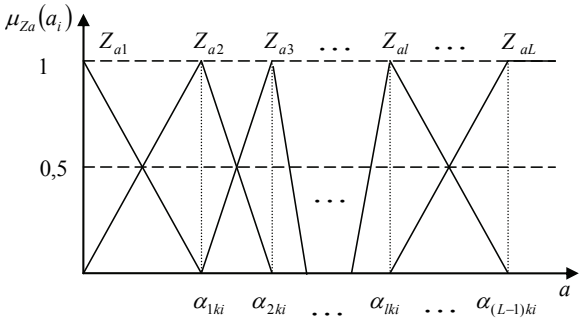
Przyjmijmy, że obciążenia a i b są zmiennymi lingwistycznymi. Rzeczywistą dziedziną fizyczną zmiennych lingwistycznych a i b jest przedział $\langle 0,\infty)$. Przyjmijmy dodatkowo, że a i b są nie tylko oznaczeniami zmiennych lingwistycznych, lecz również elementów z obszaru rzeczywistej fizycznej dziedziny zmiennych (założenie takie jest często spotykane w praktycznych zagadnieniach [73]). Zbiorem wartości lingwistycznych zmiennej lingwistycznej a jest zbiór $\{Z_{a1},...,Z_{al},...,Z_{aL}\}$, gdzie L jest liczbą zbiorów rozmytych zmiennej lingwistycznej a. Dla zmiennej lingwistycznej b zbiór wartości lingwistycznych jest następujący $\{Z_{b1},...,Z_{bm},...,Z_{bM}\}$, gdzie M jest liczbą zbiorów rozmytych zmiennej lingwistycznej b.

W procesie rozmywania obliczane są wartości stopni przynależności do zbiorów rozmytych wejść. Przyjmuje się, że poszczególne zbiory rozmyte oznaczone są w ten sam sposób, co wartości lingwistyczne. Wartość stopnia przynależności zawiera się w przedziale $\langle 0,1\rangle$ [179].

Przyjęte zostało, że funkcje przynależności dla wszystkich zbiorów rozmytych wejść będą funkcjami trójkątnymi. Funkcje te mają skończony nośnik, są odcinkowo-liniowe, dzięki czemu proces obliczania stopni przynależności nie jest czasochłonny, kształt funkcji jest opisany małą liczbą parametrów, które

mogą być strojone w procesie adaptacji i dobrze odzwierciedlają rozumienie przestrzeni lingwistycznej autora pracy.

Na rys. 2.4 przedstawiono graficzną reprezentację funkcji przynależności dla wejścia a. Analogicznie prezentują się funkcje przynależności dla wejścia b.



Rys. 2.4. Funkcje przynależności $\mu_{Zal}(a_i)$, $l = 1,...,L$, dla zbiorów rozmytych dla wejścia a

Parametry $\alpha_{1\,ki},...,\alpha_{l\,ki},...,\alpha_{(L-1)\,ki}$ określają nośnik i kształt funkcji przynależności dla wejścia a. Analogicznie opisane są funkcje przynależności dla zbiorów rozmytych wejścia b, których parametry są oznaczone $\beta_{1\,ki},...,\beta_{m\,ki},...,\beta_{(M-1)\,ki}$.

Funkcje przynależności dla zbiorów rozmytych wejścia a, którego wartość w momencie nadejścia żądania x_i wynosi a_i opisać można:

$$\mu_{Za_1}(a_i) = \begin{cases} \dfrac{\alpha_{1\,ki} - a_i}{\alpha_{1\,ki}} & dla\ 0 \le a_i < \alpha_{1\,ki} \\ 0 & dla\ \alpha_{1\,ki} \le a_i \end{cases} \tag{2.1}$$

$$\mu_{Zal}(a_i) = \begin{cases} \ldots & \\ 0 & dla\ \ 0 \le a_i \le \alpha_{(l-2)\,ki} \\ \dfrac{a_i - \alpha_{(l-2)\,ki}}{\alpha_{(l-1)\,ki} - \alpha_{(l-2)\,ki}} & dla\ \ \alpha_{(l-2)\,ki} < a_i \le \alpha_{(l-1)\,ki} \\ \dfrac{a_i - \alpha_{l\,ki}}{\alpha_{(l-1)\,ki} - \alpha_{l\,ki}} & dla\ \ \alpha_{(l-1)\,ki} < a_i < \alpha_{l\,ki} \\ 0 & dla\ \ \alpha_{l\,ki} \le a_i \\ \ldots & \end{cases}$$

$$\mu_{ZaL}(a_i)=\begin{cases}0 & dla\ \ 0\le a_i\le \alpha_{(L-2)ki}\\ \dfrac{a_i-\alpha_{(L-2)ki}}{\alpha_{(L-1)ki}-\alpha_{(L-2)ki}} & dla\ \ \alpha_{(L-2)ki}< a_i<\alpha_{(L-1)ki}\\ 1 & dla\ \ \alpha_{(L-1)ki}\le a_i\end{cases}$$

gdzie $\mu_{Zal}(a_i)$, jest funkcją przynależności l-tego zbioru rozmytego dla wejścia a. Funkcję przynależności dla zbiorów rozmytych wejścia b opisać można analogicznymi wzorami, przy czym funkcje te oznaczone zostały $\mu_{Z_{b1}}(b_i),...,\mu_{Z_{bm}}(b_i),...,\mu_{Z_{bM}}(b_i)$.

Zastosowane trójkątne funkcje przynależności zbiorów rozmytych dla obu wejść spełniają warunek podziału jedności, dlatego też $\underset{a_i\in\langle 0,\infty)}{\forall}\sum_{l=1}^{L}\mu_{Zal}(a_i)\equiv 1$ oraz $\underset{b_i\in\langle 0,\infty)}{\forall}\sum_{m=1}^{M}\mu_{Zbm}(b_i)\equiv 1$.

Na wyjściu bloku rozmywania otrzymuje się wartości stopni przynależności do poszczególnych zbiorów rozmytych, wartości te przekazywane są do bloku wnioskowania.

W następniku (konkluzji) reguł rozmytych, opisanych dalej, występuje zmienna lingwistyczna oznaczona jako t, jej rzeczywistą dziedziną jest przedział $\langle 0,\infty)$, natomiast zbiorem wartości lingwistycznych jest $\{T_1,...,T_j,...T_J\}$, gdzie J jest liczbą zbiorów rozmytych wyjścia t. Funkcje przynależności zbiorów rozmytych wyjścia t są singeltonami, wskazującymi na pewne konkretne wartości $t_{1ki},...,t_{jki},...,t_{Jki}$. Wartości te równe są czasom obsługi żądań przyporządkowanym do tej samej k_i-tej klasy dla różnych obciążeń systemu obsługi żądań HTTP. Funkcje przynależności dla wyjścia t można opisać:

$$\mu_{T_j}(t_i)=\begin{cases}1 & dla\ t_i=t_{jki}\\ 0 & dla\ t_i\ne t_{jki}\end{cases}, \tag{2.2}$$

gdzie $j=1,...,J$, t_i jest wartością zmiennej t dla i-tego żądania.

Wartości parametrów funkcji przynależności dla wejść i wyjść mogą zmieniać się w czasie, w procesie adaptacji.

Baza reguł omawianego modelu jest kompletna lingwistycznie i numerycznie, zawiera J reguł R1,…,RJ następującej postaci:

$$\begin{aligned}&\text{R1: JEŚLI } (a=Z_{a1}) \text{ I } (b=Z_{b1}) \text{ TO } (t=T_1)\\ &\text{R2: JEŚLI } (a=Z_{a1}) \text{ I } (b=Z_{b2}) \text{ TO } (t=T_2)\\ &\text{R3: JEŚLI } (a=Z_{a1}) \text{ I } (b=Z_{b3}) \text{ TO } (t=T_3)\\ &\qquad\cdots\\ &\text{Rj: JEŚLI } (a=Z_{al}) \text{ I } (b=Z_{bm}) \text{ TO } (t=T_j)\end{aligned} \tag{2.3}$$

...

RJ: JEŚLI ($a = Z_{aL}$) I ($b = Z_{bM}$) TO ($t = T_J$)

$l = 1,...,L$,

$m = 1,...,M$,

$j = 1,...,J$.

W bloku wnioskowania obliczane są stopnie spełnienia poszczególnych reguł rozmytych. Na wstępie niezbędne jest obliczenie stopnia spełnienia przesłanek dla poszczególnych reguł (stopnia zapłonu reguł). Stopień spełnienia przesłanki dla reguły Rj obliczany jest zgodnie z zależnością:

$$\mu_{Rj}(a_i, b_i) = \mathrm{T}(\mu_{Z_{al}}(a_i), \mu_{Z_{bm}}(b_i)), \tag{2.4}$$

gdzie T jest operatorem T-normy i przyjęte zostało, że będzie to operator PROD, czyli operator iloczynu. Dlatego zależność (2.4) przyjmie postać:

$$\mu_{Rj}(a_i, b_i) = \mu_{Z_{al}}(a_i) \cdot \mu_{Z_{bm}}(b_i). \tag{2.5}$$

Po obliczeniu stopni przynależności przesłanek reguł, należy obliczyć stopnie przynależności konkluzji reguł rozmytych, czyli przeprowadzić wnioskowanie (inferencje). W prezentowanym modelu wnioskowanie zostało oparte na implikacji z wykorzystaniem operatora PROD (iloczynu), zwanego również regułą Larsena. Funkcję przynależności konkluzji reguły rozmytej otrzymujemy poprzez ograniczenie pełnej funkcji przynależności konkluzji reguły do poziomu określonego przez stopień spełnienia przesłanki reguły. Zależność (2.6) prezentuje otrzymaną funkcję przynależności konkluzji reguły rozmytej:

$$\mu_{T_j^*}(t_i) = \begin{cases} \mu_{Rji} & dla\ t_i = t_{jki} \\ 0 & dla\ t_i \neq t_{jki} \end{cases}, \tag{2.6}$$

gdzie μ_{Rji} jest wartością funkcji $\mu_{Rj}(a_i, b_i)$, $j = 1,...,J$. W praktyce wartość funkcji $\mu_{T_j^*}(t_i)$ jest równa wartość $\mu_{Rj}(a_i, b_i)$. Na wyjściu bloku wnioskowania otrzymuje się wartości stopni przynależności dla przesłanek poszczególnych reguł $\mu_{R1i},...,\mu_{Rji},...,\mu_{RJi}$.

W bloku wyostrzenia (defuzyfikacji) obliczana jest „ostra" wartość na wyjściu modelu. Stosowanych jest wiele metod wyostrzania, w prezentowanym modelu wykorzystana została metoda wysokości, która jest efektywna obliczeniowo oraz dzięki której model jest „czuły" na zmiany wejść. Wynik wyostrzania otrzymywany jest w następujący sposób:

$$\hat{t}_i = \frac{\sum_{j=1}^{J}\left(t_{jki} \cdot \mu_{T_j^* i}\right)}{\sum_{j=1}^{J} \mu_{T_j^* i}}, \tag{2.7}$$

gdzie $\mu_{T_j^* i}$ oznacza wartość funkcji przynależności $\mu_{T_j^*}(t_i)$ dla $t_i = t_{jki}$. Ponieważ $\mu_{T_j^*}(t_{jki}) = \mu_{Rji}$ oraz przy spełnieniu warunku podziału jedności dla funkcji przynależności obu wejść $\sum_{j=1}^{J} \mu_{Rji} = 1$, to zależność (2.7) można przekształcić do postaci:

$$\hat{t}_i = \sum_{j=1}^{J} t_{jki}\mu_{Rji} \ . \tag{2.8}$$

W opisany powyżej sposób możliwe jest oszacowanie czasu $\hat{t}_i$ odpowiedzi na żądanie HTTP wysłane przez klienta. W procesie estymacji wykorzystywane są dane A_{ki}, B_{ki} oraz T_{ki} pobierane z modułu parametrów systemu, a stanowiące parametry funkcji przynależności dla wejść i wyjścia. Parametry te są wagami w modelu rozmyto-neuronowym, którego budowa i działanie w trybie adaptacji opisane są w kolejnym rozdziale.

2.2.2. MECHANIZM ADAPTACJI

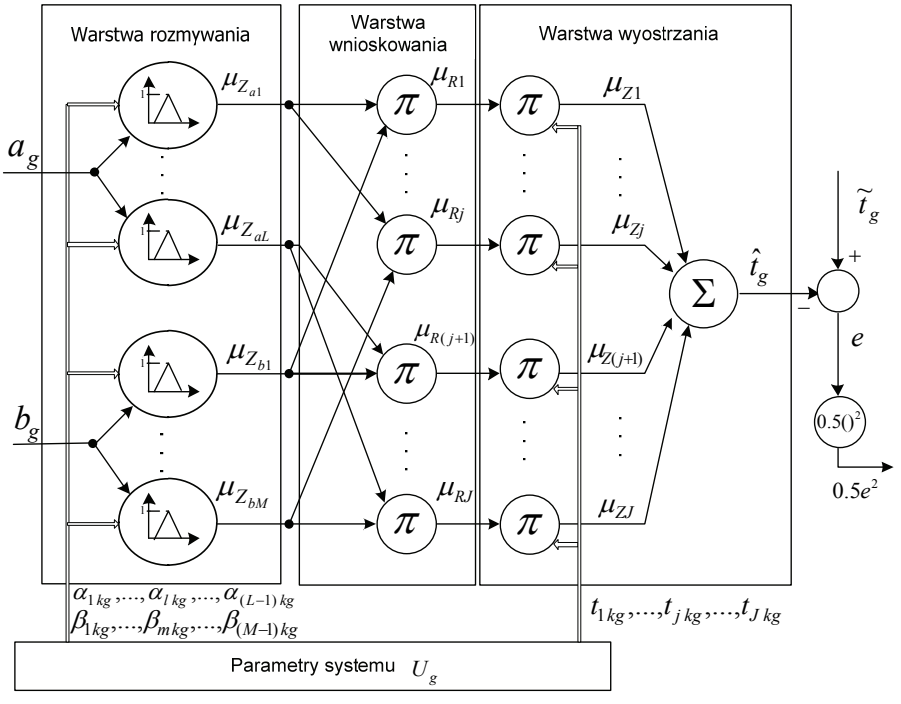
Opisywany dotychczas model rozmyty jest dość ogólny, dzięki czemu po przekształceniu w model rozmyto-neuronowy możliwe będzie zastosowanie go do szacowania czasów odpowiedzi dla systemów obsługi żądań HTTP o bardzo zróżnicowanych architekturach.

W dalszych opisach procesu adaptacji w tym rozdziale indeks dolny i zastąpiony zostanie przez indeks g, wskazując tym samym, że prowadzone będą rozważania dotyczące kolejnych żądań odnoszących się do obiektów HTTP należących do tej samej k-tej klasy, gdzie $k = k_{g-1} = k_g = k_{g+1}$. Można powiedzieć, że opisane zostanie działanie sieci rozmyto-neuronowej dla pewnej konkretnej k-tej klasy, a sieci takich jest w całym systemie K. Parametry (wagi) dla każdej z sieci przechowywane są w module parametrów systemu U_i.

Na rys. 2.5 przedstawiony jest ogólny schemat modelu rozmyto-neuronowego utworzonego w wyniku przekształcenia modelu rozmytego.

W prezentowanej sieci rozmyto-neuronowej warstwa wejściowa neuronów składa się z dwóch części, z których każda odpowiada za rozmywanie innej

wartości wejściowej. Każda z części zawiera tyle neuronów, ile jest zbiorów rozmytych dla danego wejścia, dlatego też w warstwie wejściowej jest $L+M$ neuronów. Na wyjściu każdego neuronu warstwy rozmywania otrzymywana jest wartość stopnia przynależności do poszczególnych zbiorów rozmytych wejść.



Rys. 2.5. Schemat modelu rozmyto-neuronowego umożliwiającego szacowanie czasów odpowiedzi na żądania

W warstwie wnioskowania opisywanej sieci rozmyto-neuronowej obliczane są stopnie przynależności $\mu_{T_j^*}(t_g) = \mu_{Rj}(a_g, b_g)$, $j = 1,...,J$, zmodyfikowanych funkcji przynależności do zbiorów rozmytych wyjścia. Liczba neuronów w tej warstwie odpowiada liczbie reguł w bazie reguł i wynosi J.

Struktura warstwy wyostrzenia w modelu rozmyto-neuronowym zależy od przyjętej metody wyostrzania. Jak już wspomniano wcześniej, w omawianym modelu zastosowana została metoda wysokości. Warstwa wyostrzania zawiera $J+1$ neuronów, z czego J neuronów oblicza iloczyn $t_{jkg}\mu_{Rjg}$ stopnia spełnienia j-tej reguły rozmytej, dla $j = 1,...,J$, oraz czasu t_{jkg} pobranego z modułu parametrów systemu dla k-tej klasy, natomiast jeden neuron oblicza sumę wyników uzyskanych z pozostałych neuronów warstwy. Suma ta jest wartością wyjścia, czyli szacowanym czasem $\hat{t}_g$ odpowiedzi na żądanie.

Adaptacja w modelu rozmyto-neuronowym jest realizowana poprzez strojenie jego parametrów na podstawie informacji o błędzie szacowanego czasu $\hat{t}_g$ odpowiedzi na żądanie względem rzeczywistego czasu $\tilde{t}_g$ odpowiedzi [90]. Procesowi adaptacji podlegać będą parametry zbiorów rozmytych wejść oraz wyjścia. W procesie tym wykorzystywana jest metoda wstecznej propagacji błędu wprowadzona przez J. Werbosa [174]. Algorytm adaptacji modyfikuje parametry w taki sposób, aby minimalizować wartość błędu średniokwadratowego obliczanego na podstawie wartości błędu całkowitego szacowania czasu odpowiedzi na żądanie. Błąd całkowity e_g obliczany jest jako:

$$e_g = \tilde{t}_g - \hat{t}_g \,. \tag{2.9}$$

Błąd średniokwadratowy obliczany jest w następujący sposób:

$$E_g = \frac{\left(e_g\right)^2}{2}. \tag{2.10}$$

W procesie minimalizacji błędu średniokwadratowego E_g wykorzystana została reguła najszybszego spadku opracowana przez Widrowa i Hoffa [176], zgodnie z którą parametry strojone są według zależności:

$$\delta_{(g+1)} = \delta_g - \eta \frac{\partial E_g}{\partial \delta_g}, \tag{2.11}$$

gdzie $\delta_{(g+1)}$ jest wartością zmodyfikowanego parametru δ_g, η jest wartością współczynnika uczenia, natomiast $\frac{\partial E_g}{\partial \delta_g}$ jest błędem składowym.

W celu realizacji strojenia parametrów zbiorów rozmytych wyliczane są składowe błędu średniokwadratowego ze względu na parametry, które podlegać będą strojeniu. Błędy składowe dla parametrów zbiorów rozmytych dla wyjścia wyliczane są na podstawie:

$$\frac{\partial E_g}{\partial t_{jkg}} = \left(\hat{t}_g - \tilde{t}_g\right)\frac{\partial \hat{t}_g}{\partial t_{jkg}} = \left(\hat{t}_g - \tilde{t}_g\right)\frac{\partial\left(\sum_{v=1}^{J} \mu_{Rvg} t_{vkg}\right)}{\partial t_{jkg}} = \left(\hat{t}_g - \tilde{t}_g\right)\mu_{Rjg}, \tag{2.12}$$

natomiast nowe wartości parametrów zbiorów rozmytych wyjścia wyliczane są zgodnie z zależnością:

$$t_{j\,k(g+1)} = t_{j\,kg} + \eta_t \mu_{R_j g}\left(\tilde{t}_g - \hat{t}_g\right), \tag{2.13}$$

gdzie η_t jest współczynnikiem uczenia dla parametrów rozmytych wyjścia.

W celu obliczenia błędów składowych dla parametrów zbiorów rozmytych wejścia a zależność 2.8 przekształcona jest do postaci:

$$\hat{t}_g = \sum_{m=1}^{M}\sum_{l=1}^{L} \mu_{Z_{al}}(a_g)\mu_{Z_{bm}}(b_g)t_{((m-1)L+l)kg} \ . \tag{2.14}$$

Błąd składowy dla parametru α_{lkg} wejścia a obliczany jest z zależności:

$$\begin{aligned}
&\frac{\partial E_g}{\partial \alpha_{lkg}} = (\hat{t}_g - \tilde{t}_g)\frac{\partial \hat{t}_g}{\partial \alpha_{lkg}} = \\
&= (\hat{t}_g - \tilde{t}_g)\frac{\partial\left(\sum_{m=1}^{M}\sum_{\phi=1}^{L} \mu_{Z_{a\phi}}(a_g)\mu_{Z_{bm}}(b_g)t_{((m-1)L+\phi)kg}\right)}{\partial \alpha_{lkg}} = \\
&= (\hat{t}_g - \tilde{t}_g)\sum_{m=1}^{M}\left(\mu_{Z_{bm}}(b_g)\frac{\partial\left(\sum_{\phi=1}^{L}\mu_{Z_{a\phi}}(a_g)t_{((m-1)L+\phi)kg}\right)}{\partial \alpha_{lkg}}\right) = \\
&= (\hat{t}_g - \tilde{t}_g)\sum_{m=1}^{M}\left(\mu_{Z_{bm}}(b_g)\sum_{\phi=1}^{L} t_{((m-1)L+\phi)kg}\ \frac{\mu_{Z_{a\phi}}(a_g)}{\partial \alpha_{lkg}}\right)
\end{aligned} \tag{2.15}$$

gdzie $l = 1,...,L-1$.

Uaktualnione wartości parametrów zbiorów rozmytych wejścia a można obliczyć, korzystając z zależności (2.11) oraz (2.15) w następujący sposób:

$$\alpha_{l\,k(g+1)} = \alpha_{lkg} + \eta_a(\tilde{t}_g - \hat{t}_g)\sum_{m=1}^{M}\left(\mu_{Z_{bm}}(b_g)\sum_{\phi=1}^{L} t_{((m-1)L+\phi)kg}\ \frac{\partial \mu_{Z_{a\phi}}(a_g)}{\partial \alpha_{lkg}}\right), \tag{2.16}$$

gdzie η_a jest współczynnikiem uczenia dla parametrów wyjścia a. Analogicznie obliczane są uaktualnione wartości parametrów zbiorów rozmytych wejścia b:

$$\beta_{m\,k(g+1)} = \beta_{m\,k\,g} + \eta_b(\tilde{t}_g - \hat{t}_g)\sum_{l=1}^{L}\left(\mu_{Z_{al}}(a_g)\sum_{\gamma=1}^{M} t_{((l-1)M+\gamma)kg}\ \frac{\partial \mu_{Z_{b\gamma}}(b_g)}{\partial \beta_{m\,kg}}\right), \tag{2.17}$$

gdzie $m = 1,...,M-1$, η_a jest współczynnikiem uczenia dla parametrów wyjścia a.

Na szybkość adaptacji oraz dopasowania opisywanego modelu do modelowanego systemu wpływ ma liczba zbiorów rozmytych wejść L i M oraz przyjęte wartości współczynników uczenia η_a, η_b, η_t dla wejść i wyjścia. Czym większa liczba zbiorów rozmytych wejścia, tym przyjęty model dokładniej odwzorowuje zachowanie się obiektu, który naśladuje, równocześnie jednak zmniejsza się prędkość dopasowywania do modelowanego systemu. Szybkość adaptacji zależy również w dużym stopniu od przyjętych wartości współczynników uczenia. Czym większe wartości współczynników, tym szybciej zachodzi adaptacja, ale model traci równocześnie własności generalizacji.

Po przeprowadzeniu licznych badań wstępnych dla każdej z prezentowanych dalej metod przyjęte zostało, że liczba zbiorów rozmytych dla obu wejść będzie taka sama $L = M = 4$, natomiast przyjęte wartości współczynników uczenia to $\eta_a = \eta_b = 0{,}1$ oraz $\eta_t = 0{,}3$.

2.3. PODSUMOWANIE

Przedstawiony model systemu obsługi żądań HTTP stanowi podstawę w konstrukcji pięciu spośród sześciu rozwiązań prezentowanych w monografii, dlatego też można stwierdzić, że model ten jest elastyczny. We wszystkich rozwiązaniach model umożliwia podejmowanie decyzji w czasie rzeczywistym w warunkach niepewności, gdy informacja o działaniu systemu webowego jest niedokładna lub nawet stara. Model umożliwia dopasowywanie się w sposób adaptacyjny do modelowanego systemu i nie wymaga złożonej wstępnej konfiguracji.

Przydatność oraz jakość działania modelu została zweryfikowana i zobrazowana w postaci wyników badań jakości pracy poszczególnych prezentowanych rozwiązań.

3. SYSTEMY Z KRYTERIUM CZASOWYM MINIMALIZUJĄCE CZASY ODPOWIEDZI NA ŻĄDANIA HTTP

W rozdziale przedstawione zostaną projekty systemów webowych pracujących w taki sposób, aby minimalizować czasy odpowiedzi na pojedyncze żądania HTTP. Przedstawione zostaną następujące systemy:

- LFNRD – system dystrybucji żądań HTTP w środowisku lokalnie rozmieszczonych serwerów webowych,
- GARDiB – system dystrybucji żądań HTTP w środowisku globalnie rozmieszczonych serwerów webowych z serwerami pośredniczącymi,
- GARD – system dystrybucji żądań HTTP w środowisku globalnie rozmieszczonych serwerów webowych bez serwerów pośredniczących.

3.1. SYSTEM LFNRD DYSTRYBUCJI ŻĄDAŃ HTTP W ŚRODOWISKU LOKALNIE ROZMIESZCZONYCH SERWERÓW WEBOWYCH

Stosowanie grupy lokalnie rozmieszczonych serwerów webowych jest obecnie najczęściej spotykaną metodą zwiększania wydajności serwisów webowych [45, 81]. Rozwiązanie to jest wykorzystywane przy obsłudze serwisów webowych o dużej popularności, których klienci zgromadzeni są na niezbyt rozległym obszarze geograficznym np. w obrębie jednego państwa.

Głównymi komponentami systemów webowych wieloserwerowych są [53]:

- mechanizm przekierowywania żądań HTTP – umożliwia dostarczanie żądań do serwera WWW oraz wysyłanie odpowiedzi z serwera do klienta. Od przyjętego mechanizmu przekierowywania żądań zależy architektura połączeń stosowana w klastrze,
- algorytm dystrybucji żądań – decyduje, do którego serwera będzie przekierowane żądanie,
- egzekutor – jest elementem wykonawczym, który odpowiada za przesłanie żądania użytkownika do wybranego przez algorytm dystrybucji żądań serwera zgodnie z mechanizmem przekierowywania żądań. Często też egzekutor ma za zadanie przesłanie odpowiedzi z serwera do użytkownika.

Od rodzaju zastosowanego algorytmu dystrybucji żądań i mechanizmu przekierowania w dużej mierze zależy wydajność całego serwisu webowego. Opracowując metodę dystrybucji żądań, nie tylko dla lokalnych klastrów serwe-

rów, ale również dla rozwiązań wykorzystujących serwery webowe globalnie rozmieszczone, należy precyzyjnie wskazać jak działać będzie mechanizm przekierowywania żądań oraz opracować właściwy algorytm dystrybucji.

Zostało opracowanych wiele algorytmów dystrybucji żądań dla lokalnych klastrów serwerów webowych. Algorytmy te mogą być klasyfikowane na podstawie różnych przesłanek [81]. Na wstępie algorytmy mogą być podzielone na algorytmy czwartej warstwy sieciowej oraz algorytmy siódmej warstwy sieciowej modelu ISO/OSI [53]. W pierwszej grupie algorytmów decyzja o przekierowaniu żądania podejmowana jest na podstawie nagłówka segmentu TCP (warstwy transportowej) lub nagłówka pakietu IP (warstwy sieciowej). Do algorytmów czwartej warstwy sieciowej należą: Random, RR, WRR, LL i inne. W grupie algorytmów dystrybucji żądań siódmej warstwy sieciowej decyzja podejmowana jest częściowo lub całkowicie w oparciu o zawartość nagłówka żądania HTTP. Do algorytmów tej grupy należą: SP, SI, CAP, LARD, FARD, FNRD, LFNRD i inne.

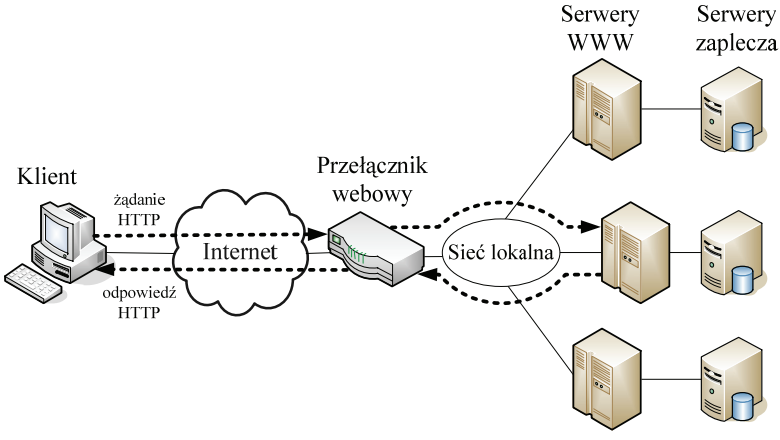
Algorytmy dystrybucji żądań mogą być również dzielone na następujące klasy: statyczne, dynamiczne i adaptacyjne. Algorytmy statyczne nie wykorzystują w swym działaniu żadnych informacji o stanie serwerów ani o żądaniu użytkownika. Dzięki temu możliwe jest bardzo szybkie podjęcie decyzji. Do algorytmów tej grupy należą: RR, Static Weighted Round-Robin (odmiana algorytmu WRR, w której wagi nie zmieniają się), Random, CAP. W algorytmach dynamicznych do podjęcia decyzji mogą być wykorzystane różne informacje o stanie obciążenia serwerów lub informacje o obiekcie, który użytkownik chciałby pobrać. Do grupy algorytmów dynamicznych należą: Dynamic Weighted Round-Robin (odmiana algorytmu WRR, w której wagi zmieniają się w trakcie pracy), LARD, SP i inne. Podjęcie decyzji o przekierowaniu żądania w przypadku algorytmów adaptacyjnych jest czasochłonne, lecz jakość tej decyzji jest lepsza niż w poprzednim rozwiązaniu. W algorytmach adaptacyjnych polityka dystrybucji żądań może się zmieniać w zależności od obciążenia serwerów lub żądań użytkowników. Do grupy algorytmów adaptacyjnych należą: AdaptLoad, FARD, FNRD, LFNRD.

W dalszej części rozdziału opisany jest system, w którym wykorzystywany jest adaptacyjny algorytm dystrybucji żądań siódmej warstwy sieciowej.

3.1.1. OPIS SYSTEMU LFNRD

W rozdziale tym przedstawiony zostanie projekt systemu LFNRD wykorzystującego w swej konstrukcji lokalnie rozmieszczone serwery webowe formujące klaster serwerów webowych. Przygotowanie projektu systemu wymagało opracowania metody dystrybucji żądań HTTP oraz algorytmu dystrybucji żądań. Prezentowana metoda nazwana została metodą LFNRD.

Na rys. 3.1 przedstawiony jest schemat systemu LFNRD.



Rys. 3.1. Schemat systemu webowego pracującego zgodnie z metodą LFNRD

W systemie LFNRD wyróżnić można następujące elementy: klientów wysyłających żądania HTTP, przełącznik webowy dystrybuujący żądania HTTP i sterujący pracą klastra, serwery WWW realizujące proces obsługi żądań HTTP z wykorzystaniem serwerów zaplecza, takich jak serwery aplikacji i serwery bazodanowe.

W proponowanym systemie klient wysyła żądanie HTTP do przełącznika webowego, ten z wykorzystaniem algorytmu dystrybucji żądań wyznacza serwer do obsługi żądania i przesyła do serwera żądanie. Serwer WWW wraz z serwerem bazodanowym obsługują żądanie. Odpowiedź HTTP wraz z żądanym obiektem HTTP przesyłana jest do przełącznika webowego, a następnie dalej do klienta. Przykładowe drogi przesyłania żądania i odpowiedzi HTTP zostały zaznaczone na rys. 3.1 przerywaną linią.

Celem działania systemu webowego pracującego zgodnie z metodą LFNRD jest wybranie spośród serwerów WWW w klastrze tego serwera, który oferuje najkrótszy czas odpowiedzi dla każdego pojedynczego żądania HTTP oddzielnie. Decyzja o przekierowaniu żądań oraz nadzór nad obsługą żądania realizowany jest w specjalnie skonstruowanym przełączniku webowym nazywanym również dalej przełącznikiem LFNRD.

Zdefiniujmy pojęcia czasu odpowiedzi w systemie LFNRD.

Definicja 3.1. Czasem odpowiedzi $\widetilde{t}_i$ na żądanie HTTP x_i w systemie LFNRD jest czas liczony od momentu rozpoczęcia wysyłania żądania HTTP przez przełącznik webowy do serwera WWW do momentu otrzymania odpowiedzi na żądania w całości w przełączniku webowym.

W skład czasu odpowiedzi na żądanie wchodzi czas transmisji żądania HTTP z przełącznika do serwera WWW, czas przebywania żądania na serwerze WWW, serwerze aplikacji i bazodanowym oraz czas transmisji obiektu HTTP z serwera WWW do przełącznika webowego.

Przyjmuje się, że każdy z serwerów WWW w lokalnym klastrze może obsłużyć każde z żądań HTTP dopuszczalnych do obsługi w ramach danego serwisu webowego i każdy z serwerów obsłuży żądanie w jednakowy sposób, dotyczy to zarówno żądań o obiekty statyczne, jak i obiekty dynamiczne.

Obsługa żądań realizowana jest zgodnie ze polityką First Come First Served (FCFS), a więc w takiej kolejności, w jakiej żądania przychodzą do kolejki wejściowej przełącznika webowego. Wszystkie żądania HTTP traktowane są jednakowo. Przełącznik webowy oraz serwery WWW obsługują protokół HTTP w wersji 1.0 oraz 1.1.

Podstawowym elementem sterującym działaniem systemu LFNRD jest przełącznik LFNRD, dla którego założenia oraz projekt przedstawione są poniżej.

Przyjmuje się następujące dodatkowe oznaczenia:

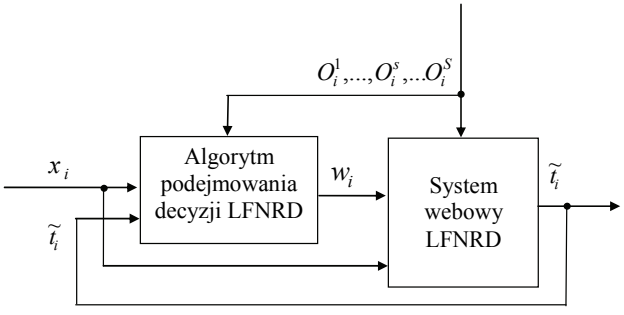
- O_i^s – obciążenie s-tego realizatora (serwera webowego) w i-tym momencie,
- s – indeks serwera webowego, $s \in \{1,...,S\}$,
- S – liczba realizatorów (serwerów webowych) wykonujących obsługę żądań HTTP,
- $\hat{t}_i^s$ – szacowany czas odpowiedzi dla s-tego serwera na i-te żądanie HTTP,
- w_i – decyzja, numer realizatora (serwera) wybranego do obsługi i-tego żądania, $w_i \in \{1,...,S\}$,
- ar_i^s – obciążenie s-tego serwera webowego, jego wartość jest równa liczbie równocześnie obsługiwanych żądań HTTP przez serwer,
- bd_i^s – obciążenie s-tego serwera webowego, jego wartość równa jest liczbie równocześnie obsługiwanych żądań HTTP dotyczących obiektów dynamicznych,
- y_i^s – odpowiedź s-tego serwera na i-te żądanie.

Zadanie projektowe: należy opracować projekt przełącznika webowego, który dla każdego przychodzącego żądania x_i wyznaczy na podstawie wiedzy o obciążeniach serwerów webowych $O_i^1,...,O_i^S$ oraz wiedzy o przeszłych czasach odpowiedzi na żądania $\tilde{t}_1,...,\tilde{t}_{i-1}$ serwer webowy w_i, $w_i \in \{1,...,S\}$, dla którego szacowany czas odpowiedzi $\hat{t}_i^s$, $s \in \{1,...,S\}$, na żądanie x_i jest najmniejszy, przy założeniach $\underset{x_i \in X \wedge k,l \in \{1,...,S\}}{\forall} y_i^k = y_i^l$.

Projektowany przełącznik webowy powinien działać tak, aby minimalizować czasy odpowiedzi na żądania oraz aktualizować swoją wiedzę na temat systemu na podstawie przeszłych czasów odpowiedzi, przy założeniach, że

wszystkie serwery mogą obsłużyć wszystkie żądania obsługiwane w systemie webowym.

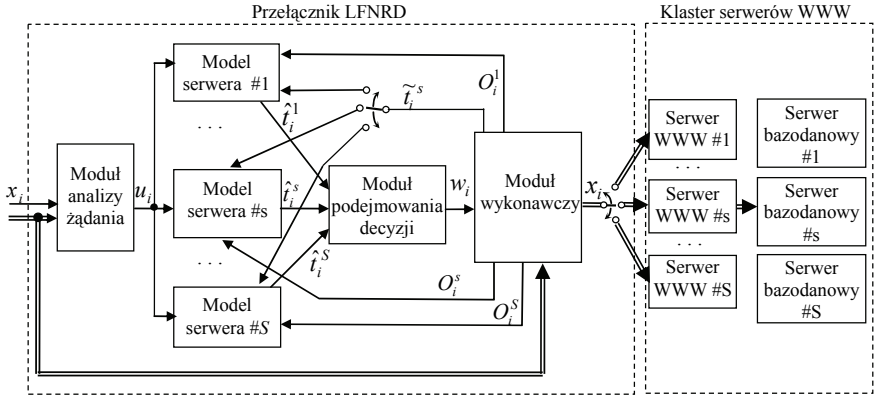
Na rys. 3.2 przedstawiony został ogólny schemat prezentujący proces podejmowania decyzji w systemie LFNRD. Zgodnie z przyjętą koncepcją w systemie, a dokładniej w przełączniku webowym, wyróżniony jest odpowiedni algorytm podejmowania decyzji, który realizuje decyzje zgodnie z przedstawionymi powyżej założeniami.



Rys. 3.2. Schemat prezentujący proces podejmowania decyzji w systemie LFNRD

3.1.2. PRZEŁĄCZNIK LFNRD

Na rys. 3.3 przedstawiony jest schemat przełącznika LFNRD. Przełącznik składa się z następujących głównych elementów: modułu analizy żądania, modeli serwerów, modułu podejmowania decyzji i modułu wykonawczego.



Rys. 3.3. Schemat przełącznika LFNRD wraz z klastrem serwerów WWW

Moduł analizy żądania w przełączniku LFNRD analizuje nadchodzące żądanie x_i i pobiera z niego adres żądanego obiektu u_i.

W module modelu serwera szacowany jest czas odpowiedzi na żądanie x_i. Przełącznik zawiera S modeli serwerów, czyli tyle, ile serwerów WWW zawiera klaster. Każdy model serwera jest przyporządkowany dokładnie do jednego serwera WWW i szacuje czas odpowiedzi $\hat{t}_i^s$ przyporządkowanego serwera, $s = 1,...,S$. Model serwera jest modelem systemu obsługi żądań HTTP. Model ten szacuje czas odpowiedzi na podstawie adresu u_i żądanego obiektu i obciążenia O_i^s serwera WWW. W procesie adaptacji model serwera wykorzystuje również zmierzony czas $\tilde{t}_i$ odpowiedzi na żądanie x_i.

Moduł podejmowania decyzji wybiera serwer, który obsłuży żądanie x_i. Decyzją w_i jest numer wybranego serwera, $w_i \in \{1,...S\}$. Wybierany jest ten serwer, dla którego szacowany czas odpowiedzi jest najkrótszy:

$$w_i : \hat{t}_i^{w_i} = \min\left\{\hat{t}_i^s : s = 1,...,S\right\}. \tag{3.1}$$

Moduł wykonawczy jest egzekutorem i fizycznie przekazuje żądanie x_i do wybranego serwera o numerze w_i. Moduł wykonawczy nadzoruje proces przesyłania danych pomiędzy klientem a serwerem WWW, dlatego też posiada informacje o rodzaju obsługiwanych żądań i czasie odpowiedzi. Po zakończeniu obsługi żądania informacja o zmierzonym czasie obsługi $\tilde{t}_i$ jest przekazywana do modułu modelu serwera odpowiadającego serwowi, który obsłużył żądanie. Dodatkowo moduł wykonawczy przekazuje informacje $O_i^1,...,O_i^s,...,O_i^S$ do modułów modeli serwerów o obciążeniu poszczególnych serwerów. Wektor $O_i^s = \left[a_i^s, b_i^s\right]$ charakteryzuje obciążenie s-tego serwera WWW oraz bazodanowego i jest odpowiednikiem obciążenia O_i w opisywanym wcześniej modelu systemu obsługi żądań HTTP. Przyjmuje się, że składowa obciążenia $a_i^s = ar_i^s$, gdzie ar_i^s jest obciążeniem s-tego serwera WWW i jego wartość jest równa liczbie równocześnie obsługiwanych żądań HTTP przez serwer, natomiast $b_i^s = bd_i^s$, gdzie bd_i^s jest liczbą równocześnie obsługiwanych żądań HTTP dotyczących jedynie obiektów dynamicznych. Moduł wykonawczy pozyskuje na bieżąco wartości obciążeń poszczególnych serwerów i są to dane dostępne w ramach przełącznika webowego.

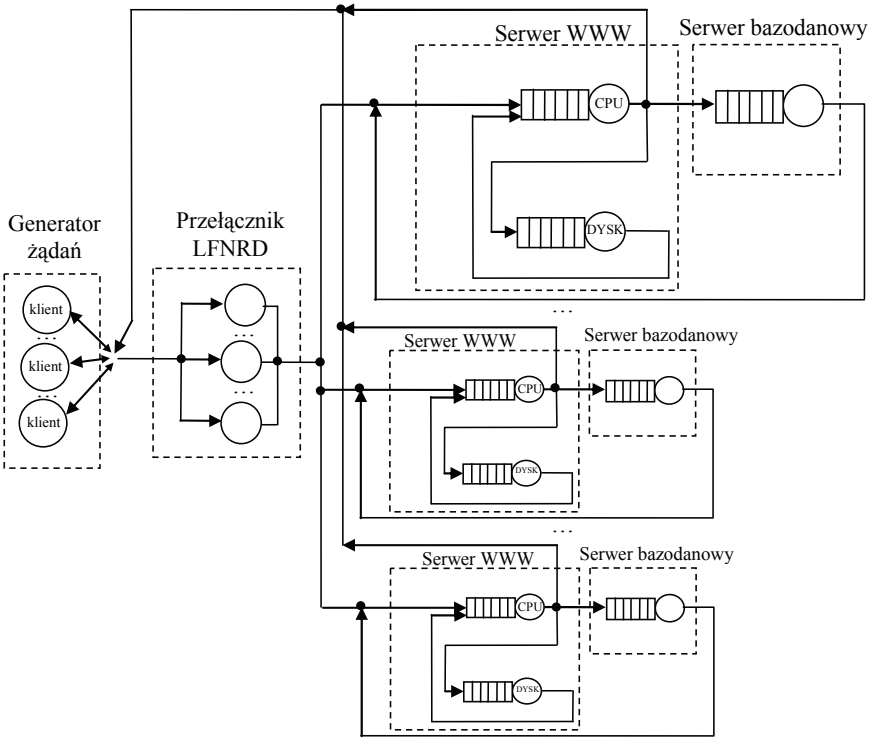
3.1.3. BADANIA DOTYCZĄCE SYSTEMU LFNRD

W celu określenia jakości pracy systemu LFNRD działającego zgodnie z metodą LFNRD przeprowadzone zostały badania symulacyjne. Wykonane zostało odpowiednie stanowisko badawcze. Do wykonania programu symulacyjnego wykorzystany został pakiet CSIM19 [70], który jest zbiorem bibliotek napisanych w języku C++, umożliwiających tworzenie zorientowanych na procesy modeli symulacyjnych wykorzystujących zdarzenia dyskretne. Pakiet ten jest dobrze znany, przetestowany, uznany i jest często stosowany w symulacji środowisk webowych [6, 28, 56, 130, 187, 193, 194, 217, 218].

Schemat przyjętego modelu symulacyjnego zaprezentowany został na rys. 3.4. Rysunek prezentuje sieć kolejkową, w której poszczególne elementy odpowiadają zaimplementowanym modułom w programie symulacyjnym. W skład symulatora wchodziły następujące moduły: moduł generatora żądań, moduł przełącznika LFNRD, moduł serwera WWW, moduł serwera bazodanowego.

W badaniach symulacyjnych przyjęty został znany z literatury i szeroko stosowany model generatora żądań HTTP [11, 42, 52, 56, 69, 130, 199], który umożliwił modelowanie pracy klientów. Dzięki zastosowanemu modelowi generowany strumień żądań był zgodny z obserwowanym w Internecie ruchem żądań HTTP, charakteryzującym się dużą zmiennością, wybuchowością (nagłym, gwałtownym wzrostem liczby żądań HTTP) i samopodobieństwem. Strumień taki może być generowany z wykorzystaniem rozkładów długoogonowych takich jak Pareto oraz logarytmiczno-normalny [50, 198, 201, 203]. Zalety wskazanego modelu potwierdzone zostały w [177]. W tab. 3.1 zestawione zostały rozkłady prawdopodobieństwa oraz wartości parametrów rozkładów wykorzystanych w generatorze żądań HTTP.

W trakcie działania generator żądań tworzył w czasie jednej sekundy zadaną liczbę nowych symulacyjnych klientów. Klient rozpoczynał pobieranie strony od pozyskania szkieletu HTML strony, następnie pobierane były kolejne obiekty zagnieżdżone na stronie. Rozmiary szkieletów strony generowane był zgodnie ze złożonym rozkładem logarytmiczno-normalnym oraz Pareto, natomiast wielkości obiektów zagnieżdżonych w stronie generowane były zgodnie z rozkładem logarytmiczno-normalnym. Średnia wielkość strony wynosiła około 80 KB. Między momentami zakończenia pobierania poprzedniego obiektu i rozpoczęcia pobierania następnego występowała przerwa, jest to czas namysłu przeglądarki – moment, gdy prawdziwe przeglądarki internetowe analizują otrzymane dane i przygotowują kolejne żądanie. Po pobraniu wszystkich obiektów strony, których liczba modelowana była zgodnie z rozkładem Pareto, następowała przerwa w pracy klienta na czas namysłu użytkownika. Czas ten w świecie rzeczywistym występuje, gdy użytkownik czyta zawartość strony internetowej. W kolejnym kroku klient pobierał następną stronę. Liczba pobranych stron (odsłon) modelowana była zgodnie z rozkładem odwrotnym Gausowskim. Po pobraniu wszystkich stron w ramach sesji klient symulacyjny kończył działanie.



Rys. 3.4. Schemat modelu symulacyjnego systemu webowego LFNRD

Tabela 3.1

Rozkłady modelujące pracę klientów internetowych

Kategoria	Rozkład	Parametry rozkładu
Liczba odsłon na sesje	Odwrotny Gausowski	μ=3,86; λ=9,46
Czas namysłu użytkownika	Pareto	α=1,4; k=1
Liczba obiektów na stronie	Pareto	α=1,33; k=2
Czas namysłu przeglądarki	Weibul	α=7,640; σ=1,705
Rozmiar szkieletu strony HTML część zasadnicza ogon	 Logarytmiczno-normalny Pareto	 μ=7,630; σ=1,001 α=1; k=10240
Rozmiar obiektu zagnieżdżonego	Logarytmiczno-normalny	μ=8,215; σ=1,46

Serwis WWW w programie symulacyjnym mógł obsługiwać żądania statyczne (dotyczące obiektów statycznych) bądź żądania dynamiczne (dotyczące obiektów dynamicznych). Żądania dynamiczne obsługiwane były przez serwer WWW oraz serwer bazodanowy, natomiast żądania statyczne tylko przez serwer WWW, z wykorzystaniem pamięci podręcznej serwera.

W badaniach przyjęto, że 80% żądań dotyczyło zasobów statycznych, natomiast 20% zasobów dynamicznych [203].

Żądania dynamiczne zostały podzielone na trzy klasy [56]:

- mocno intensywne, mocno obciążające serwer bazodanowy, stanowiące 1% żądań dynamicznych,
- średnio intensywne, średnio obciążające serwer bazodanowy, stanowiące 14% żądań dynamicznych,
- mało intensywne, mało obciążające serwer bazodanowy, stanowiące 85% żądań dynamicznych.

Przyjęte zostało, że sumaryczna wielkość obiektów statycznych udostępnianych przez serwis wynosiła 400 MB.

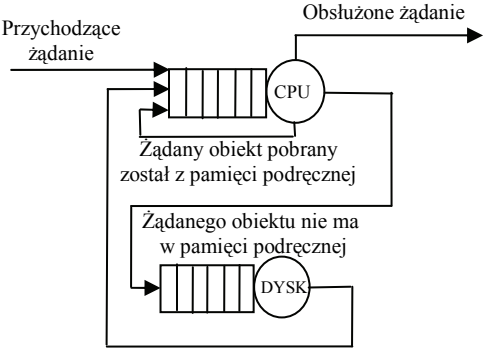
Klienci w programie symulacyjnym przesyłali żądania do modułu przełącznika LFNRD. Przyjęte zostało, że łącze między klientami a przełącznikiem ma nieskończoną przepustowość. Wiąże się to z faktem, że obecnie w sieciach lokalnych wykorzystuje się technologie, dzięki którym czas przesyłania żądań i odpowiedzi HTTP w sieciach lokalnych jest znacząco mniejszy od czasu przygotowania danych w serwerze webowym. Sieć lokalna, ze względu na dużą przepustowość, nie może stać się „wąskim gardłem" w systemie.

W module przełącznika wszystkie żądania obsługiwane były współbieżnie. Ponieważ czas obsługi żądania przez przełącznik jest mały w stosunku do czasu odpowiedzi na żądanie, przyjęto, że czas obsługi przez przełącznik jest do pominięcia. Zarówno żądania jak i odpowiedzi na żądania HTTP przekazywane były do klientów za pośrednictwem przełącznika webowego.

W celu porównania jakości pracy klastra działającego pod kontrolą algorytmu LFNRD z działaniem klastra pracującego pod kontrolą innych algorytmów referencyjnych oraz algorytmów najczęściej stosowanych w rozwiązaniach przemysłowych, w module przełącznika zaimplementowane zostały również algorytmy: FNRD, LARD, CAP, WRR, RR. Algorytm WRR przydzielał różną liczbę żądań użytkowników do serwerów WWW w zależności od przyjętych wag. Wagi dla poszczególnych serwerów wyliczane były z określonym interwałem na podstawie liczby aktywnie obsługiwanych żądać HTTP przez poszczególne serwery. Miara ta dobrze odzwierciedla obciążenie serwera i jest często stosowana w algorytmach dystrybucji żądań [28, 56, 130]. Algorytm przekierowywał większą liczbę żądań do serwerów mniej obciążonych.

Na rys. 3.5 zaprezentowany został model symulacyjny serwera WWW. Moduł serwera WWW składał się z: procesora, dysku twardego oraz pamięci podręcznej [5, 12, 69, 133].

Procesor oraz dysk twardy modelowane były jako systemy kolejkowe z pojedynczą kolejką oraz jednym stanowiskiem obsługi. Czasy obsługi żądań dobrane zostały w badaniach eksperymentalnych dla serwera z procesorem Intel Pentium 4 z zegarem 2 GHz oraz dyskiem twardym Seagate ST340810A 80GB IDE. Na serwerze zainstalowany został system operacyjny Linux Fedora Core 6 oraz oprogramowanie serwera WWW Apache 2.2.4 wraz z modułami PHP 5.1.



Rys. 3.5. Schemat modelu symulacyjnego serwera WWW

Czasy związane z obsługą żądań opisać można w następujący sposób [204, 217]:

- Czasy obsługi żądania na procesorze:
  - czas nawiązania połączenia TCP: $S_{TCP}^{CPU} = 0{,}10097\,\text{ms}$,
  - czas analizy żądania i przygotowania odpowiedzi HTTP: $S_{AP}^{CPU} = 0{,}14533\,\text{ms}$,
  - czas transmisji danych: $S_{T}^{CPU}(z) = z \cdot 0{,}004291\,\text{ms}$, gdzie z jest wielkością żądanego obiektu wyrażoną w KB,
- czas obsługi żądania na dysku modelowany był zgodnie z zależnością:

$$S^{D}(z) = \begin{cases} 4{,}5 \text{ dla } z \in <0{,}128> \\ 0{,}03813125 \cdot z - 0{,}3808 \text{ dla } z \in (128, \infty) \end{cases} \tag{3.2}$$

gdzie czas obsługi na dysku S^{D} wyrażony jest w milisekundach. Podane parametry pracy serwera uwzględniają fakt, że żądania obsługiwane były na procesorze wielordzeniowym z wieloma poziomami pamięci podręcznych.

Zostało przyjęte za [11], że pamięć podręczna serwera WWW w modelu serwera pracować będzie zgodnie z algorytmem Last Recently Used (LRU) (obiekty najdawniej używane usuwane są z pamięci w pierwszej kolejności).

Serwer bazodanowy modelowany był jako system kolejkowy z jedną kolejką do zasobu i stanowiskiem obsługi. Czasy obsługi żądań dynamicznych na stanowisku obsługi serwera bazodanowego modelowane były zgodnie z rozkładem hiperwykładniczym za [69]. Czasy obsługi żądań dynamicznych zależały od rodzaju żądania (mocno, średnio oraz mało intensywne). W tab 3.2 zestawione są przyjęte czasy obsługi żądań dynamicznych na serwerze bazodanowym.

Tabela 3.2

Czasy obsługi żądań na serwerze bazodanowym

Typ żądania dynamicznego	Średni czas obsługi [ms]
Mało intensywne	10
Średnio intensywne	50
Mocno intensywne	100

Badania symulacyjne prowadzone były z wykorzystaniem programu symulacyjnego, którego konstrukcja opisana została powyżej. Badane były następujące miary odzwierciedlające jakość obsługi żądań przez klaster serwerów webowych:

- średni czas odpowiedzi na żądanie HTTP,
- 95-percentyl czasu odpowiedzi strony internetowej.

Czas odpowiedzi strony internetowej liczony był jako suma czasów odpowiedzi wszystkich obiektów strony $T_{strony} = \sum_{n=1}^{N} \tilde{t}_i$, gdzie N jest liczbą obiektów zagnieżdżonych w stronie plus jeden obiekt szkieletu HTML strony, $\tilde{t}_n$ jest czasem odpowiedzi na żądanie HTTP dotyczące n-tego obiektu strony.

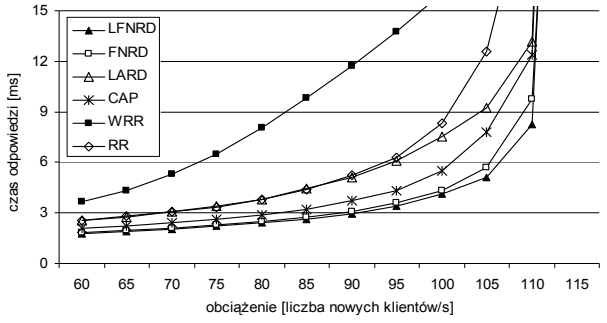
O ile miara, jaką jest średni czas odpowiedzi na żądanie HTTP obrazuje jakość działania zaproponowanego systemu LFNRD, którego celem działania jest minimalizowanie czasu odpowiedzi na pojedyncze żądania, o tyle użytkownicy pracujący z przeglądarkami internetowymi nie będą w stanie zarejestrować faktu szybszego pobrania pojedynczych obiektów. Użytkownicy zarejestrują czas pobrania całej strony internetowej składającej się z większej grupy obiektów. Dlatego też 95 percentyl czasu odpowiedzi strony jest miarą, która lepiej prezentuje działanie systemu obserwowane z punktu widzenia użytkowników.

Wskazane miary jakości obsługi żądań dobrze odzwierciedlają jakość pracy systemu webowego oraz są bardzo często stosowane zarówno praktycznie do oceny systemu webowego [9, 10] jak i w badaniach naukowych [1, 3, 4, 87, 173].

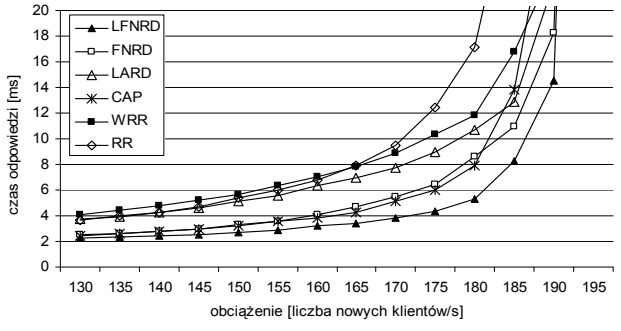
Badania przeprowadzone zostały dla czterech wariantów konfiguracji serwerów w klastrze. W pierwszym wariancie wykorzystane zostały trzy homogeniczne serwery WWW i bazodanowe, których opis i konfiguracja przedstawiona została powyżej (konfiguracja oznaczona została na rysunkach Hom3s). W drugim wariancie klaster składał się z trzech serwerów WWW i bazodanowych, przy czym jeden zestaw serwerów miał wydłużone wszystkie czasy obsługi żądań na poszczególnych stanowiskach obsługi o 33% (oznaczenie na rysunkach Het1s/2s). W trzecim wariancie wykorzystanych zostało pięć homogenicznych serwerów WWW i bazodanowych (oznaczenie na rysunkach Hom5s). W czwartym wariancie klaster składał się z pięciu serwerów WWW i bazodanowych, z czego dwa zestawy miały wydłużone wszystkie czasy obsługi żądań na poszczególnych stanowiskach obsługi o 33% (oznaczenie na rysunkach Het2s/3s).

Na rys. 3.6 przedstawione zostały średnie czasy odpowiedzi na żądania w funkcji obciążenia (liczby nowych klientów generowanych na sekundę), natomiast na rys. 3.7 przedstawiony został 95 percentyl czasu odpowiedzi w funkcji obciążenia.

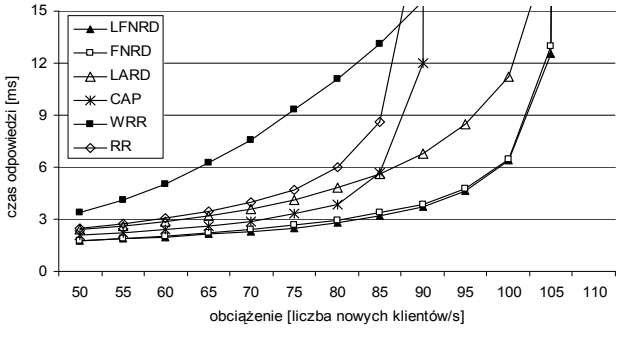
a)



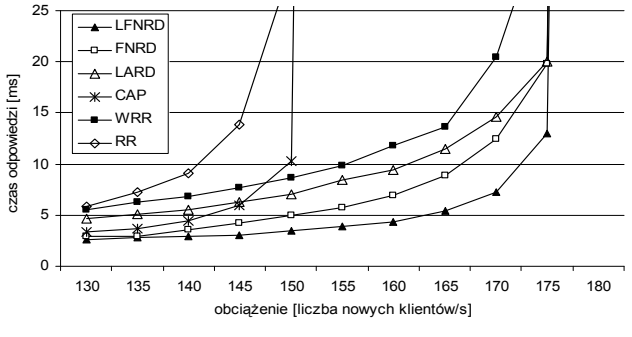
b)



c)

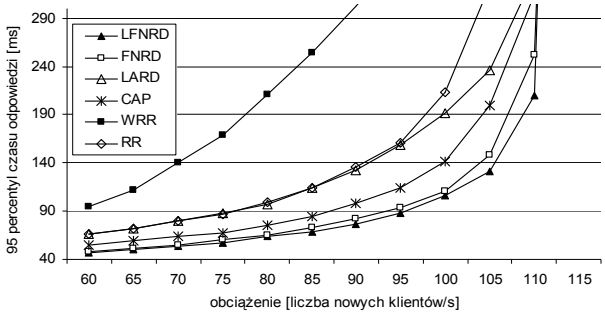


d)

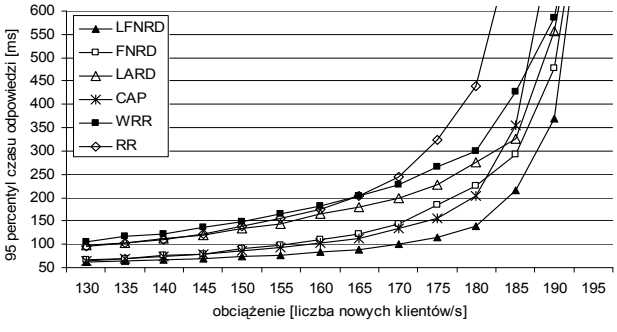


Rys. 3.6. Średnie czasy odpowiedzi na żądanie w funkcji liczby nowych klientów dla wariantów: a) Hom3s, b) Hom5s, c) Het1s/2s, d) Het2s/3s

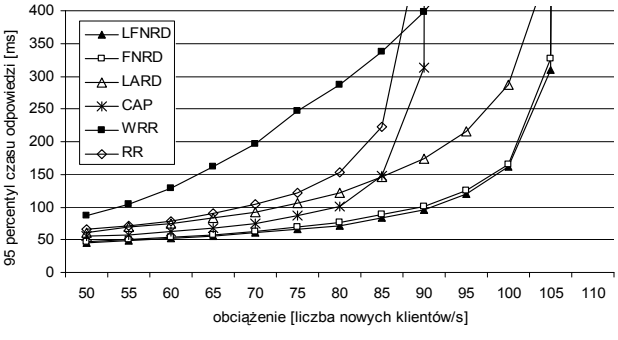
a)



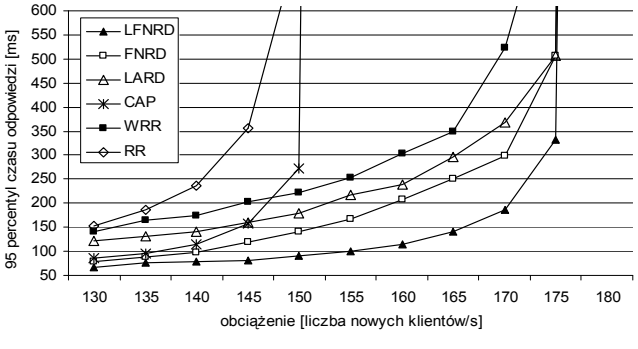
b)



c)



d)



Rys. 3.7. 95 percentyl czasu odpowiedzi strony w funkcji liczby nowych klientów dla wariantów: a) Hom3s, b) Hom5s, c) Het1s/2s, d) Het2s/3s

Z wykresów zaprezentowanych na rys. 3.6 oraz 3.7 wynika, że najkrótsze średnie czasy odpowiedzi na żądania oraz najmniejszą wartość 95 percentyla czasu odpowiedzi strony uzyskane zostały dla algorytmu LFNRD, zarówno w przypadku klastrów homogenicznych jak i heterogenicznych. Dobre wyniki dla tego algorytmu uzyskane zostały dla mniejszego trzyserwerowego klastra oraz większego pięcioserwerowego. W przypadku klastrów serwerów heterogenicznych wyniki dla algorytmu LFNRD były znacznie lepsze niż dla pozostałych algorytmów, czasy odpowiedzi przy dużym obciążeniu były dla tego algorytmu prawie dwukrotnie krótsze przy większym obciążeniu niż dla pozostałych algorytmów.

Dla algorytmu FNRD uzyskiwane były podobne wyniki jak dla LFNRD w klastrze z mniejszą liczbą serwerów. Dość krótkie czasy odpowiedzi uzyskane zostały również dla algorytmu CAP dla homogenicznych konfiguracji klastra.

Jak zostało wskazane powyżej, na podstawie wyników badań symulacyjnych, opłacalne jest stosowanie złożonego algorytmu dystrybucji żądań. Należy jednak jeszcze rozważyć, czy przełącznik webowy pracujący pod kontrolą algorytmu LFNRD może stać się „wąskim gardłem" całego systemu webowego.

Zostały przeprowadzone badania określające wydajność przełącznika webowego, pracującego pod kontrolą algorytmów wykorzystanych w opisanych powyżej badaniach. W tab. 3.3 zestawione zostały wyniki badań przeprowadzonych na serwerze z dwoma procesorami Intel Dual-Core Xeon 5160 2,4 GHz z pamięcią RAM 4 GHz. Jako miara wydajności przyjęta została liczba podjętych decyzji o przełączeniu żądania HTTP na sekundę. Badania przeprowadzone zostały dla niezoptymalizowanego oprogramowania i wykonane dla czerech równocześnie pracujących wątków.

Tabela 3.3

Liczba decyzji podjętych w ciągu sekundy i średni czas podjęcia decyzji dla różnych algorytmów dystrybucji żądań

Algorytm	Średnia liczba decyzji podjętych w ciągu sekundy	Średni czas podjęcia decyzji
LFNRD	616 tys.	0,001623 ms
FNRD	4,5 mln	0,000222 ms
LARD	12 mln	0,000083 ms
CAP	12 mln	0,000083 ms
WRR	15 mln	0,000066 ms
RR	20 mln	0,000050 ms

Z otrzymanych wyników badań zaprezentowanych w tab. 3.3 wynika, że przełącznik webowy pracujący pod kontrolą algorytmu LFNRD jest najmniej wydajny. Jednak średni czas podjęcia decyzji jest kilka rzędów mniejszy niż czas obsługi żądania przez serwer webowy. Uzyskane wyniki prezentowane na rys. 3.6 oraz 3.7 wskazują, że zastosowanie algorytmu LFNRD może znacząco skrócić czas obsługi żądań HTTP nawet o kilka milisekund, równocześnie podnosząc wydajność serwisu. Dodatkowo szacuje się, że przy odpowiedniej optymalizacji kodu algorytmu LFNRD możliwe byłoby podniesienie wydajności

przełącznika nawet trzykrotnie. Jak wynika z informacji podanych przez [123] w lutym 2009 r. liczba odwiedzin serwisu Google.pl w miesiącu styczniu 2009 wyniosła 4217mln, można stąd szacować, że średnia liczba żądań obsługiwanych na sekundę wyniosła około 12200, co jest liczbą znacznie mniejszą niż liczba podejmowanych decyzji z zastosowaniem algorytmu LFNRD.

Na zakończenie obliczona zostanie pesymistyczna czasowa złożoność obliczeniowa. Wyznaczenie decyzji zgodnie z algorytmem LFNRD wiąże się z wyszukaniem żądanego obiektu i wskazaniem jego klasy, oszacowaniem czasu odpowiedzi w sieci rozmyto-neuronowej dla każdego z S serwerów webowych, wyznaczeniem decyzji, a na zakończenie z aktualizacją wartości parametrów systemu. Pesymistyczna złożoność obliczeniowa wyszukania elementu wynosi dla drzewa binarnego $O(\log N)$, gdzie przyjmijmy, że N jest liczbą obiektów w serwisie, czas wyznaczenia klasy żądanego obiektu jest stały. Czas oszacowania czasu odpowiedzi w sieci rozmyto-neuronowej jest stały dla każdego z serwerów, dlatego też złożoność obliczeniowa zależy jedynie od liczby serwerów S i w konsekwencji $O(S)$. Czas aktualizacji parametrów systemu jest stały. Podsumowując, złożoność obliczeniowa algorytmu LFNRD wynosi $O(\log N + S)$. Wynika stąd, że odpowiednio wykonany, zoptymalizowany, dedykowany przełącznik webowy LFNRD mógłby być wykorzystany w systemach webowych obsługujących dużą liczbę użytkowników.

W przeglądowym opracowaniu [53] zaproponowane zostało zastosowanie kilku przełączników webowych pracujących w odpowiedniej hierarchii w przypadku budowy dużego serwisu webowego, w którym pojedynczy przełącznik webowy mógłby stać się potencjalnym „wąskim gardłem". W serwisie tym wszystkie żądania HTTP kierowane są do wydajnego przełącznika webowego pracującego w czwartej warstwie sieciowej, ten natomiast przekierowuje żądania do przełączników webowych warstwy siódmej. Rozwiązanie to gwarantuje z jednej strony bardzo dużą wydajność grupy przełączników, z drugiej wysoką jakość podjętych decyzji przez przełączniki pracujące w siódmej warstwie sieciowej modelu ISO/OSI. Opisane tu rozwiązanie umożliwiałoby zastosowanie przełączników LFNRD w bardzo dużych serwisach webowych.

Jak zostało wskazane powyżej, zastosowanie systemu i metody LFNRD może znacząco podnieść jakość obsługi klientów. Dalsze badania nad systemem oraz metodą LFNRD zaprezentowane zostały w [208, 218, 220].

3.2. SYSTEM GARDIB DYSTRYBUCJI ŻĄDAŃ HTTP W ŚRODOWISKU GLOBALNIE ROZMIESZCZONYCH SERWERÓW WEBOWYCH Z SERWERAMI POŚREDNICZĄCYMI

Opisany w rozdz. 3.1 system dystrybucji żądań HTTP w lokalnym klastrze serwerów z powodzeniem może być wykorzystany do obsługi popularnych serwisów webowych, których użytkownicy rozmieszczeni są w pewnym ograniczonym obszarze np. w jednym kraju lub jednym kontynencie. Jeżeli jednak bardzo liczna grupa użytkowników rozproszona jest na dużym obszarze geograficznym np. na kilku kontynentach, wtedy stosuje się globalnie rozproszone systemy webowe.

Globalnie rozproszony system webowy składa się z dowolnej liczby podsystemów, nazwanych na potrzeby tego opracowania serwisami lokalnymi. Globalnie rozproszone podsystemy występują pod jednym adresem mnemonicznym np. www.google.com. Poszczególne serwisy lokalne, rozmieszczone mogą być w różnych lokalizacjach geograficznych. Serwis lokalny może być klastrem lokalnie rozmieszczonym lub pojedynczym serwerem webowym.

Obecnie stosowane globalnie rozmieszczone systemy webowe wykorzystują różne mechanizmy dystrybucji żądań HTTP, jednym z najpopularniejszych jest mechanizm wykorzystujący odpowiednio działający serwer DNS (ang. Domain Name System), który na żądanie klienta tłumaczy adres mnemoniczny serwisu WWW na adres IP. Serwer DNS z odpowiednio zaimplementowanym algorytmem podejmowania decyzji może wskazać adres IP serwisu lokalnego, który obsłuży żądania HTTP klienta [16, 67, 127, 129, 134, 162, 180].

Stosowanie serwerów DNS w dystrybucji żądań w globalnie rozproszonych systemach webowych ma wiele korzyści, należą do nich między innymi łatwość w budowie, prostota konfiguracji oraz niski koszt tworzenia całego systemu webowego. Jedną z głównych wad takiego rozwiązania jest brak kontroli nad dystrybucją żądań klienta. Klient po otrzymaniu adresu IP z serwera DNS wysyła wszystkie żądania do tego samego serwisu lokalnego. Innym rozwiązaniem proponowanym natomiast w tym rozdziale jest zastosowanie serwerów pośredniczących, do których klienci wysyłają żądania HTTP, a serwery pośredniczące przekierowują dalej żądania do serwisów lokalnych realizujących obsługę żądań [54, 57, 76, 137, 146, 222].

Decyzje o przekierowaniu żądań w globalnie rozproszonych systemach webowych podejmowane mogą być na dwóch poziomach. Na pierwszym poziomie – globalnym, podejmowane są decyzje o przekierowaniu żądań do odpowiednich serwisów lokalnych, natomiast na drugim poziomie – lokalnym, podejmowane są decyzje o przekierowaniu żądań wewnątrz serwisów lokalnych.

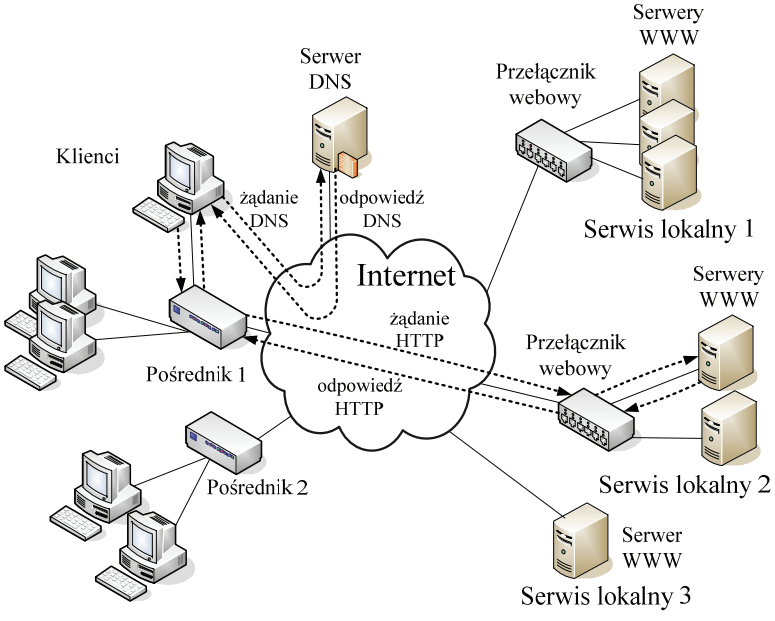
Opisany w rozdziale system GARDiB umożliwia podejmowanie decyzji polegających na wyznaczeniu odpowiedniego serwisu lokalnego w oparciu

o oszacowany z punktu widzenia użytkownika końcowego czas odpowiedzi na żądanie, co jest nowe w tego typu rozwiązaniach.

W przedstawionym opisie systemu zaproponowane zostało zastosowanie serwerów pośredniczących, dających dużą kontrolę nad obsługą każdego pojedynczego żądania HTTP. Przedstawiona została ogólna koncepcja metody oraz projekt serwera pośredniczącego. Dodatkowo opisana została budowa stanowiska laboratoryjnego oraz wyniki badań wraz z dyskusją nad wynikami.

3.2.1. OPIS SYSTEMU GARDiB

System webowy GARDiB składa się z następujących elementów: serwisów lokalnych, serwerów pośredniczących nazywanych również dalej pośrednikami, serwera DNS, klientów wysyłających żądania HTTP. Na rys. 3.8 przedstawiony został ogólny widok omawianego systemu. Sposób działania i funkcjonalności poszczególnych elementów w systemie determinuje metoda GARDiB opracowana dla systemu GARDiB.



Rys. 3.8. Schemat globalnej dystrybucji żądań HTTP w systemie GARDiB

Serwis lokalny jest lokalnym klastrem serwerów webowych lub pojedynczym serwerem webowym. Przyjmuje się, że każdy serwis lokalny może obsłużyć każde żądanie HTTP z puli żądań udostępnianych w serwisie.

Klient może wysyłać żądania do serwisów lokalnych tylko za pośrednictwem serwerów pośredniczących.

Serwery pośredniczące powinny być umiejscowione na krawędzi Internetu, czyli na styku szkieletowej sieci rozległej z sieciami miejskimi lub lokalnymi. Pośrednik powinien równocześnie znajdować się w niedużej odległości od znaczącej grupy użytkowników korzystających z serwisu.

Przyjmuje się, że w systemie mogą być wykorzystane standardowe serwery webowe, aplikacji i serwery bazodanowe oraz że działanie systemu nie wymaga modyfikacji standardowych protokołów sieciowych, a w szczególności protokołów HTTP, TCP i IP.

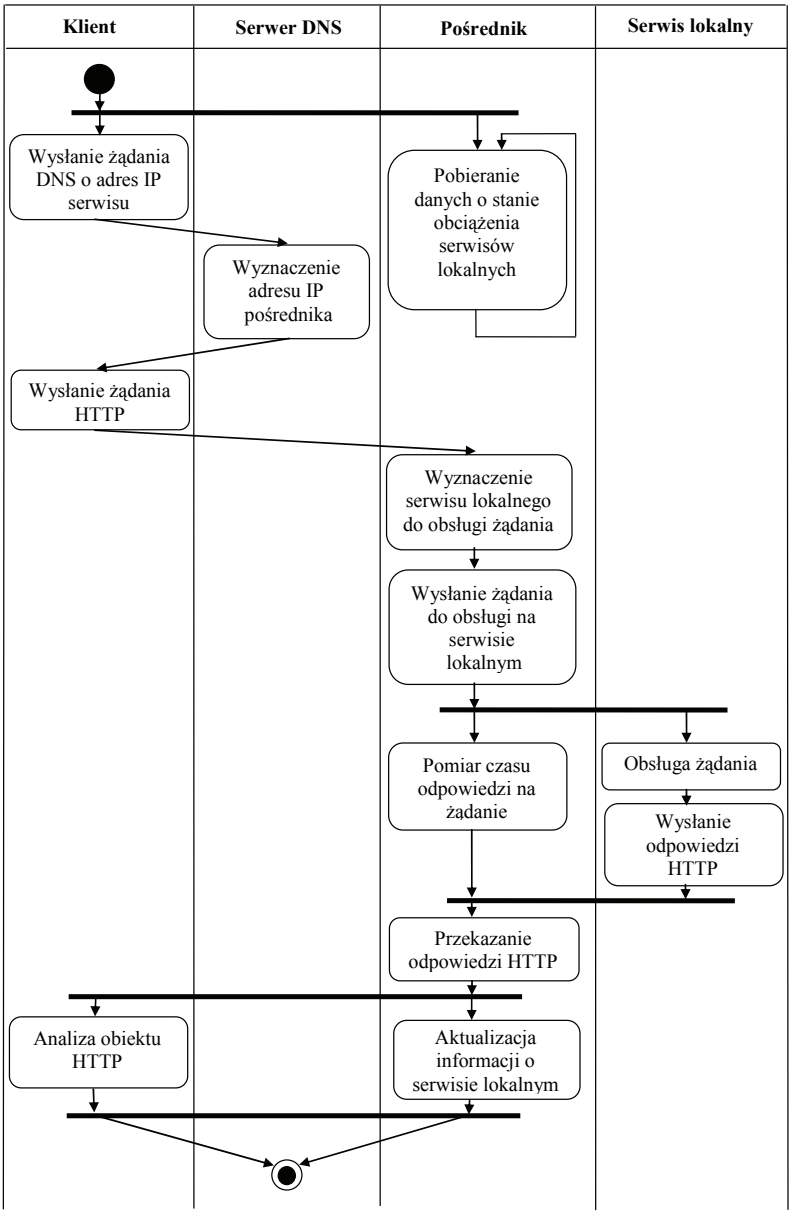
Na rys. 3.9 przedstawiony został diagram czynności, obrazujący kolejne akcje podejmowane w celu wykonania obsługi żądania klienta.

Na początku klient za pośrednictwem lokalnego serwera DNS wysyła żądanie DNS do autorytatywnego serwera DNS obsługującego domenę danego serwisu. Odpowiednio działający autorytatywny serwer DNS lub też system takich serwerów wchodzących w skład systemu GARDiB zwraca odpowiedź DNS zawierającą adres IP serwera pośredniczącego, który znajduje się w najbliżej odległości geograficznej od klienta. Rozwiązania tak działających systemów DNS znane są z literatury [2, 66, 91, 94] i nie będą stanowiły treści dalszych rozważań. Klient po otrzymaniu adresu IP pośrednika przesyła do niego żądanie HTTP. Pośrednik, posiadając wiedzę o obciążeniu poszczególnych serwisów lokalnych i obciążeniu łączy na drodze do serwisów lokalnych, wybiera z wykorzystaniem algorytmu GARDiB serwis lokalny, dla którego oszacowany czas odpowiedzi jest najkrótszy. Następnie pośrednik wysyła do wybranego serwisu lokalnego żądanie HTTP otrzymane od klienta. Serwis lokalny obsługuje żądanie. Jeśli serwis lokalny jest klastrem serwerów webowych, to żądanie w pierwszej kolejności dociera do przełącznika webowego, a ten podejmuje decyzje o przekierowaniu żądania i przesyła żądanie do odpowiedniego serwera WWW. W momencie, gdy odpowiedź HTTP jest gotowa, to przygotowany obiekt HTTP jest przesyłany do pośrednika. Pośrednik mierzy czas odpowiedzi na żądanie HTTP i po otrzymaniu odpowiedzi HTTP w całości przesyła ją do klienta, a następnie modyfikuje w procesie adaptacji swoje informacje o serwisie lokalnym, który obsłużył żądanie.

Czas odpowiedzi na żądanie HTTP w systemie GARDiB definiowany jest w następujący sposób.

Definicja 3.2. Czasem odpowiedzi $\tilde{t}_i$ na żądanie HTTP x_i w systemie GARDiB jest czas, który upływa od momentu podjęcia przez serwer pośredniczący decyzji o alokacji żądania do serwisu lokalnego, do momentu uzyskania odpowiedzi y_i w całości w serwerze pośredniczącym.

W skład czasu odpowiedzi na żądanie wchodzi: czas transmisji żądania HTTP z serwera pośredniczącego do serwisu lokalnego, czas obsługi żądania w serwisie lokalnym oraz czas transmisji obiektu HTTP z serwisu lokalnego do serwera pośredniczącego.



Rys. 3.9. Diagram czynności obsługi żądania HTTP w systemie webowym GARDiB

Ponieważ pośrednik umieszczony jest przeważnie w niedużej odległości od klienta, zatem czas przesyłania danych między klientem a pośrednikiem jest znacznie krótszy od czasu odpowiedzi na żądanie HTTP, dlatego też przyjmuje

się, że czas odpowiedzi w systemie GARDiB jest czasem rejestrowanym przez użytkownika.

Najważniejszym elementem w systemie GARDiB sterującym działaniem całego systemu jest serwer pośredniczący GARDiB. Od jego konstrukcji i sposobu działania zależy jakość obsługi żądań.

W celu sformułowania zadania projektowego przyjmuje się następujące dodatkowe oznaczenia:

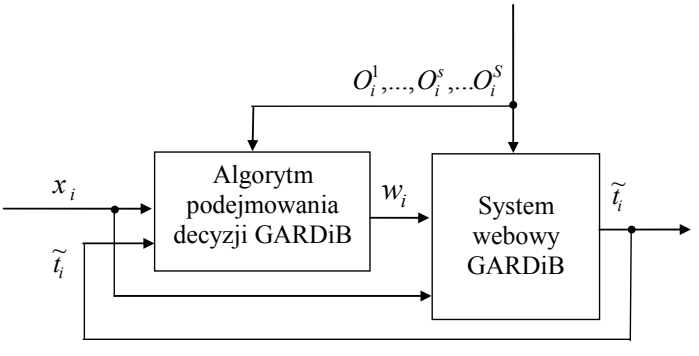
arl_i^s – obciążenie s-tego serwisu lokalnego, jego wartość równa jest liczbie żądań równocześnie obsługiwanych w serwisie lokalnym,

btt_i^s – obciążenie sieci rozległej na kierunku od pośrednika do s-tego serwisu lokalnego, jego wartość równa jest czasowi transferu próbnego obiektu od serwisu lokalnego do pośrednika.

Zadanie projektowe: Należy opracować projekt serwera pośredniczącego GARDiB, który dla każdego przychodzącego żądania x_i wyznaczy na podstawie wiedzy o obciążeniach serwisów lokalnych $O_i^1,...,O_i^S$ oraz wiedzy o przeszłych czasach odpowiedzi na żądania $\tilde{t}_1,...,\tilde{t}_{i-1}$ serwis lokalny w_i, $w_i \in \{1,...,S\}$, dla którego szacowany czas odpowiedzi $\hat{t}_i^s$, $s \in \{1,...,S\}$, na żądanie x_i jest najmniejszy, przy założeniach $\underset{x_i \in X \wedge k,l \in \{1,...,S\}}{\forall} y_i^k = y_i^l$.

Projektowany przełącznik webowy powinien umożliwiać minimalizowanie czasów odpowiedzi na żądania oraz aktualizację wiedzy na temat systemu na podstawie przeszłych czasów odpowiedzi.

Rys. 3.10 przedstawia ogólny schemat procesu podejmowania decyzji w systemie GARDiB. Schemat ten jest podobny do schematu zaprezentowanego dla systemu LFNRD.



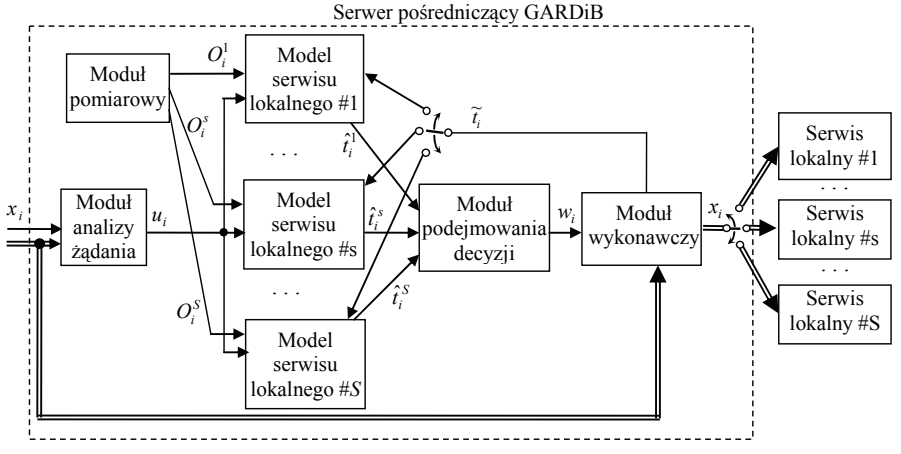
Rys. 3.10. Schemat prezentujący proces podejmowania decyzji w systemie GARDiB

3.2.2. SERWER POŚREDNICZĄCY GARDiB

Konstrukcja serwera pośredniczącego GARDiB podobna jest do konstrukcji przełącznika webowego LFNRD. Podobnie jak w przypadku przełącznika LFNRD celem działania pośrednika jest wybranie do obsługi tego serwisu lokalnego, dla którego szacowany czas odpowiedzi jest najkrótszy.

Wszystkie żądania obsługiwane są przez pośrednika z jednakowym priorytetem współbieżnie lub równolegle. Inicjalizacja obsługi żądań realizowana jest według strategii FCFS.

Na rys. 3.11 przedstawiony jest schemat serwera pośredniczącego. Serwer pośredniczący zawiera: moduł analizy żądania, moduł pomiarowy, moduł podejmowania decyzji, moduł wykonawczy oraz modele serwisów lokalnych, których liczba odpowiada liczbie serwisów lokalnych pracujących w globalnie rozproszonym systemie webowym.



Rys. 3.11. Schemat serwera pośredniczącego GARDiB

Moduł analizy żądania pobiera z żądania x_i adres u_i żądanego obiektu. Każdy **model serwisu lokalnego** przyporządkowany jest do jednego serwisu lokalnego pracującego w systemie. Modele serwisów lokalnych szacują czasy odpowiedzi $\hat{t}_i^s$, $s = 1,...,S$, na żądanie x_i. Oszacowane czasy odpowiedzi są przekazywane do modułu podejmowania decyzji. Model serwisu lokalnego jest modelem systemu obsługi żądań HTTP opisanym w rozdz. 2.1. Czas odpowiedzi szacowany jest na podstawie adresu u_i żądanego obiektu oraz obciążenia O_i^s serwisu lokalnego. Po zakończeniu obsługi żądania moduł wykonawczy przekazuje modelowi serwisu odpowiadającego serwisowi, który obsłużył żądanie,

zmierzony czas odpowiedzi $\tilde{t}_i$ na żądanie. Na podstawie tego czasu modyfikowane są parametry modelu serwisu lokalnego.

Moduł podejmowania decyzji wybiera serwis lokalny, który obsłuży żądanie x_i. Decyzją w_i, $w_i \in \{1,...S\}$, jest numer serwisu lokalnego. Wybierany jest ten serwis lokalny, dla którego szacowany czas odpowiedzi jest najkrótszy:

$$w_i : \hat{t}_i^{w_i} = \min\left\{\hat{t}_i^{s} : s = 1,...,S\right\} . \tag{3.1}$$

Moduł wykonawczy przesyła żądanie x_i do wybranego serwisu lokalnego. Po zakończeniu obsługi przez serwis moduł ten przesyła odpowiedź do klienta. Moduł wykonawczy mierzy również czas odpowiedzi $\tilde{t}_i$ na żądanie HTTP.

Moduł pomiarowy pozyskuje obciążenia serwisów lokalnych oraz mierzy stan obciążenia sieci komputerowej na kierunkach od pośrednika do poszczególnych serwisów lokalnych. Moduł ten przekazuje informacje $O_i^1,...,O_i^s,...,O_i^S$ do modułów modeli serwerów. Obciążenie s-tego serwisu lokalnego $O_i^s = \left[a_i^s, b_i^s\right]$ charakteryzowane jest przez liczbę żądań równocześnie obsługiwanych w serwisie lokalnym $a_i^s = arl_i^s$ oraz czas transferu $b_i^s = btt_i^s$ próbnego obiektu od serwisu lokalnego do pośrednika. Czas transferu próbnego obiektu jest miarą obciążenia na kierunku od pośrednika do s-tego serwisu lokalnego. W celu uzyskania wartości czasu transferu pośrednik wysyła co określony interwał czasowy żądanie HTTP do serwisu lokalnego, a w odpowiedzi serwis lokalny wysyła próbny obiekt o wielkości około 20 KB. Wielkość obiektu została tak dobrana, aby obiekt mógł być przesłany w kilku pakietach IP, a równocześnie, by przesyłanie danych nie wpłynęło na stan obciążenia samej sieci. Obsługa omawianego żądania HTTP powinna być realizowana z wysokim priorytetem w serwisie lokalnym tak, aby na czas obsługi w serwisie lokalnym był znikomo mały w stosunku do czasu przesyłania danych. Czas transferu btt_i^s liczony jest od momentu wysłania pierwszego bajtu żądania przez pośrednika do momentu otrzymania ostatniego bajtu z odpowiedzią.

Liczba żądań arl_i^s stanowiących obciążenie serwisu lokalnego jest przekazywana do pośrednika co określony interwał czasowy. Przyjmuje się, że interwał zarówno dla miary obciążenia serwisów lokalnych jak i dla miary obciążenia sieci powinien wynosić 0,5 s, dzięki czemu pośrednicy posiadać będą w miarę aktualne dane o obciążeniu, a proces pozyskiwania wartości obciążenia nie wpłynie negatywnie na działanie serwisów lokalnych.

3.2.3. BADANIA DOTYCZĄCE SYSTEMU GARDiB

Przeprowadzone zostały badania, które umożliwiły określenie jakości pracy systemu webowego pracującego z wykorzystaniem metody i algorytmu GARDiB. Badania miały charakter symulacyjny. Program symulacyjny wykonany został z wykorzystaniem pakietu CSIM19.

W skład symulatora wchodziły następujące moduły: moduł generatora żądań, moduł serwera pośredniczącego, moduł Internetu i moduł serwisu lokalnego.

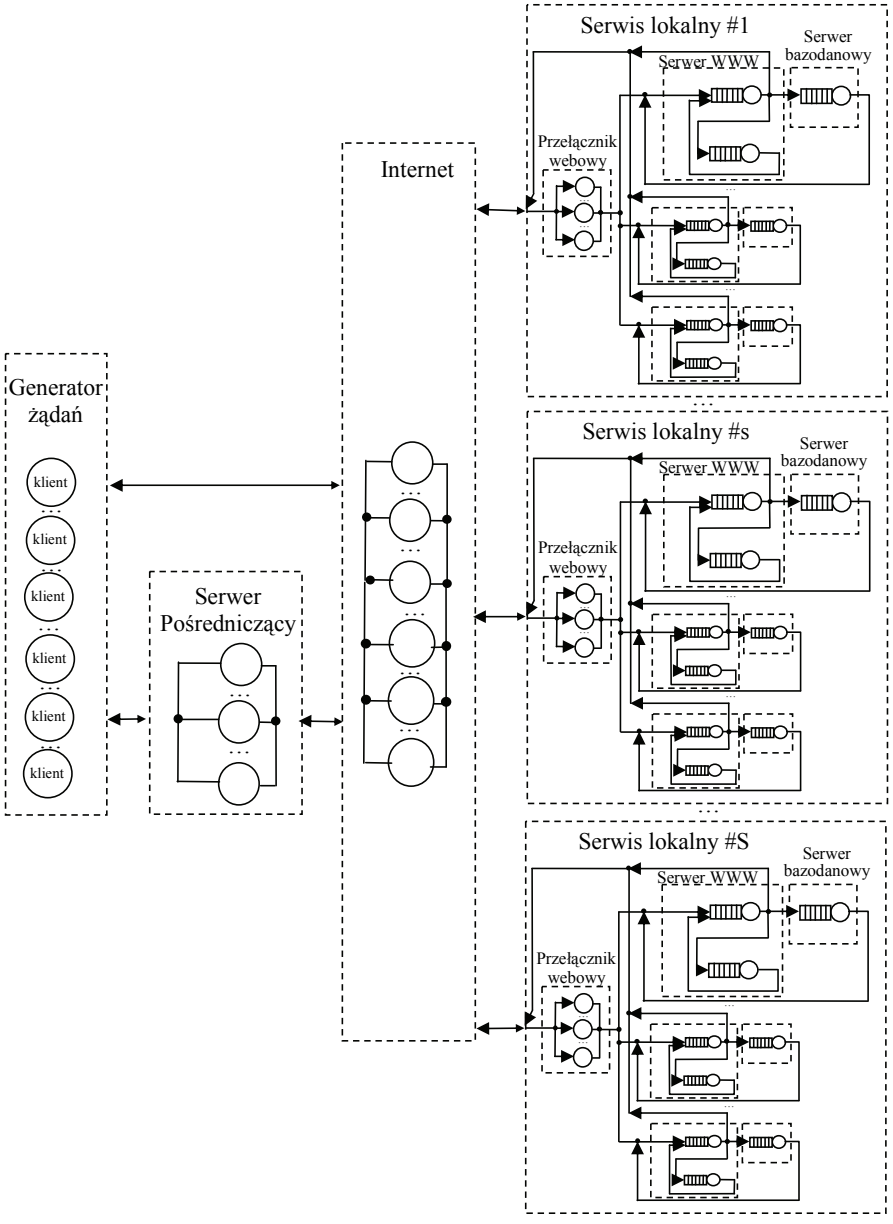
Na rys. 3.12 zaprezentowany został schemat modelu symulacyjnego wykorzystanego w konstruowaniu symulatora.

Moduł serwisu lokalnego składał się z modułów: przełącznika webowego, modułów serwerów WWW i modułów serwerów bazodanowych. Moduł serwisu lokalnego działał w podobny sposób jak symulator klastra serwerów WWW opisany w rozdz. 3.1.3. We wszystkich badaniach moduły przełączników webowych wykorzystywały algorytm LFNRD w dystrybucji żądań w serwisach lokalnych. Również moduł generatora żądań został skonstruowany tak, jak zostało to opisane w rozdz. 3.1.3, dlatego też szczegółowo zostaną omówione jedynie moduły serwera pośredniczącego i Internetu.

Moduł serwera pośredniczącego odbierał żądania HTTP dostarczane przez moduł generatora żądań. Zadanie tego modułu polegało na wyznaczeniu serwisu lokalnego, zgodnie z wybranym algorytmem dystrybucji żądań oraz przekazaniu za pośrednictwem modułu Internetu żądania do serwisu lokalnego. W module pośrednika zaimplementowane zostały cztery algorytmy dystrybucji żądań, są to:

- Global Adaptive Request Distribution with Broker (GARDiB),
- Round-Robin (RR),
- Weighted Round-Robin (WRR), w badaniach wykorzystane zostały dwie odmiany tego algorytmu, z których każda korzystała z innych miar obciążenia. Algorytmy zostały odpowiednio nazwane:
    - Weighted Round-Robin Load (WRR_L), w tej odmianie algorytmu wagi były dynamicznie modyfikowane na podstawie informacji o liczbie żądań aktywnie obsługiwanych przez serwis lokalny,
    - Weighted Round-Robin Transfer (WRR_T), wagi modyfikowane były na podstawie informacji o obciążeniu sieci Internet na kierunkach od pośrednika do serwisów lokalnych.

W obu odmianach algorytmu WRR, w trakcie badań, wagi były modyfikowane co 0,5 s czasu symulacyjnego na podstawie aktualnych informacji o obciążeniu.



Rys. 3.12. Schemat modelu symulacyjnego w badaniach nad metodą i algorytmem GARDiB

Moduł pośrednika oprócz przekierowywania żądań pobierał co określony interwał czasowy informacje o stanie obciążenia serwisów lokalnych i sieci Internet.

Moduł Internet przekazywał żądania od pośrednika oraz od generatora żądań do serwisów lokalnych, przekazywał również odpowiedzi na żądania w przeciwną stronę. Wszystkie przesłane przez ten moduł informacje były przekazywane z opóźnieniem charakterystycznym dla sieci Internet [118, 214]. Opóźnienie to nazwane *czasem transferu* jest obliczane na podstawie wzoru [126]:

$$czas\_transferu = RTT + \frac{wielkosc\_obiekt\_HTTP + wielkosc\_naglowka\_odpowiedzi\_HTTP}{efektywna\_przepustowosc}, \tag{3.4}$$

gdzie RTT (ang. Round Trip Time) jest czasem przesłania pojedynczego pakietu IP od pośrednika do serwisu lokalnego i z powrotem. Wielkość nagłówka odpowiedzi HTTP jest zmienna, średnio wynosi 290B [126]. Efektywna przepustowość jest liczbą bajtów danych (w warstwie protokółu aplikacji modelu ISO/OSI) przesłanych w jednostce czasu.

Wartości czasu RTT oraz efektywnej przepustowości zostały pozyskane po przeprowadzeniu badań. Z wybranych serwisów internetowych pobierany był plik dokumentu tekstowego RFC 1945 o wielkości 47 KB. Plik pobierany był z wykorzystaniem oprogramowania wget [175], natomiast dedykowane do tego celu oprogramowanie zapisywało wartości czasu RTT oraz efektywnej przepustowości.

Jako źródła dokumentu RFC wybrane zostały mało obciążone serwery WWW w ośrodkach akademickich posiadających szerokopasmowe łącze do Internetu. Wybrane zostały następujące ośrodki: w Polsce (PL) – Interdyscyplinarne Centrum Modelowania Matematycznego i Komputerowego na Uniwersytecie Warszawskim, w Holandii (NL) – Organizacja SURFnet, w Australii (AU) – The Australian National University, Canberra, w Brazyli (BR) – Universidade Estadual Paulista w San Paulo, w USA – Massachusetts Institute of Technology.

Przyjęte zostało, że serwisy lokalne oraz pośrednik podłączone będą do sieci Internet łączami szerokopasmowymi o dużej przepustowości, które nie będą stanowiły „wąskich gardeł" systemu. Sieć Internet jest siecią rozległą o złożonej topologii, która będzie modelowana jako zasób bez kolejki, dla którego czas obsługi na stanowisku obsługi obliczany jest zgodnie z zależnością (3.4).

W trakcie działania programu symulacyjnego generator żądań wysyłał żądania od określonej części użytkowników do pośrednika, którego działania były obserwowane i dla którego czas odpowiedzi na żądania był mierzony. Pozostała, ustalona część żądań, wysyłana była z udziałem modułu Internet do poszczególnych serwisów lokalnych. Ten dodatkowy ruch, nie kontrolowany przez obserwowanego pośrednika, odzwierciedlał ruch pochodzący z pozostałych serwerów pośredniczących.

Dla oceny metody GARDiB badane były, podobnie jak dla metody LFNRD (rozdz. 3.1.3), średnie czasy odpowiedzi na żądania HTTP oraz 95 percentyl czasu odpowiedzi strony internetowej.

Badania prowadzone były dla czterech różnych wariantów konfiguracji serwisów lokalnych. Pierwsze dwa badania przeprowadzone zostały dla trzech serwisów lokalnych umiejscowionych w Holandii, USA i Australii. W każdej z lokalizacji umieszczone zostały jednakowe, trzy serwery WWW i bazodanowe. W wariancie pierwszym do każdego serwisu lokalnego wysyłane były żądania pochodzące od 25% użytkowników (można przyjąć, że żądania te przesyłane były z nieobserwowanych serwerów pośredniczących). Również 25% użytkowników wysyłało żądania do obserwowanego serwera pośredniczącego umiejscowionego w Polsce. Prezentowane dalej wyniki badań dotyczą użytkowników obsługiwanych za pośrednictwem obserwowanego serwera pośredniczącego. Pierwszy wariant badań oznaczony został NL/3/25%; USA/3/25%; AU/3/25%; pośrednik/25% (w oznaczeniach wymieniana jest lokalizacja, liczba serwerów i procent obciążenia kierowanego na serwis lokalny).

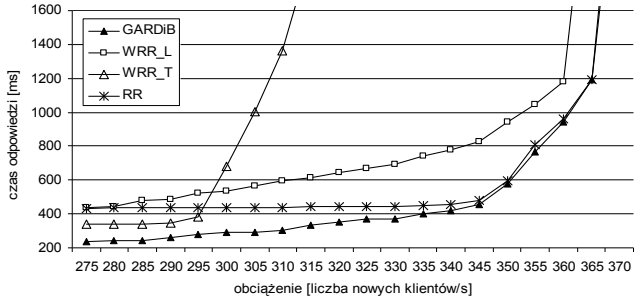
W wariancie drugim serwisy lokalne zostały obciążone nierównomiernie w ten sposób, że serwis lokalny znajdujący się najdalej (w sensie sieciowym) w Australii był obciążony najmniej i otrzymał 20% żądań, serwis w USA 25%, natomiast najbliższy pośrednikowi serwis lokalny w Holandii otrzymał 30%. Wariant wykorzystany w eksperymencie drugim oznaczony został NL/3/30%; USA/3/25%; AU/3/20%; pośrednik/25%.

W eksperymencie trzecim wykorzystane zostały również trzy serwisy lokalne z tym, że każdy z serwisów zawierał różną liczbę serwerów WWW. Najwięcej serwerów przydzielonych zostało najdalszemu serwisowi lokalnemu, natomiast najbliższemu najmniej. Obciążenie serwisów lokalnych zostało tak dobrane, aby na każdy serwer WWW przypadało 10% obciążenia pochodzącego z nieobserwowanych serwerów pośredniczących. Konfiguracja klastrów i ich obciążenie przedstawiało się następująco: NL/1/10%; USA/2/20%; AU/3/30%; pośrednik/40%.

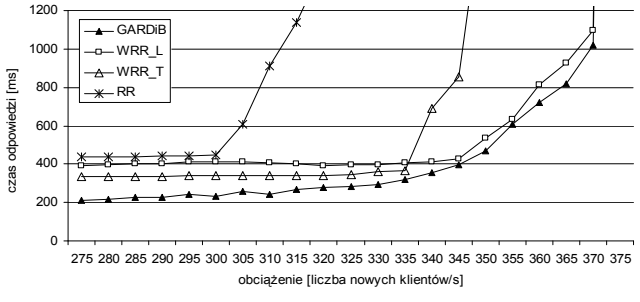
W eksperymencie czwartym wykorzystanych zostało pięć serwisów lokalnych, z których każdy zawierał po trzy serwery WWW i bazodanowe. Na wszystkie serwisy lokalne i pośrednika skierowane zostało obciążenie po 16,666% żądań. Wariant wykorzystany w eksperymencie oznaczony został następująco: NL/3/16,666%; USA/3/16,666%; AU/3/16,666%; BR/3/16,666%; PL/3/16,666%; pośrednik/16,666%.

Na rys. 3.13 przedstawione zostały średnie czasy odpowiedzi na żądanie w funkcji liczby nowych klientów, natomiast na rys. 3.14 95 percentyl czasu odpowiedzi strony w funkcji liczby nowych klientów dla wszystkich eksperymentów.

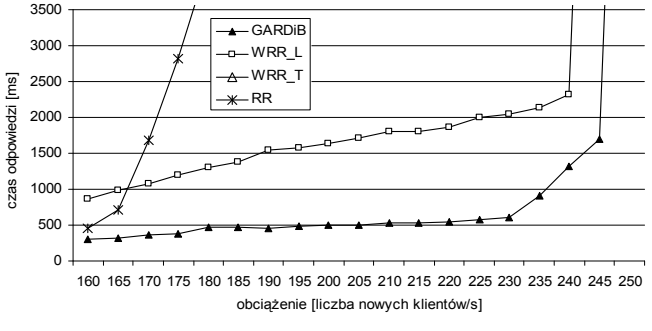
a)



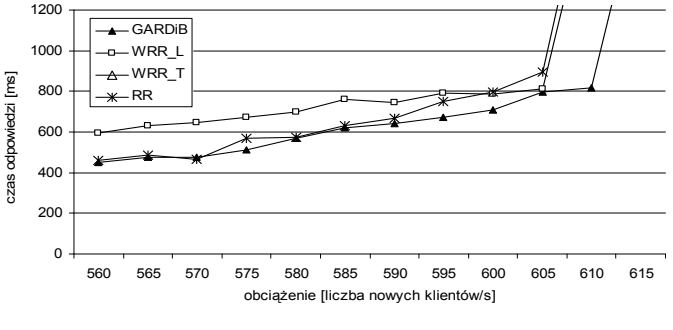
b)



c)

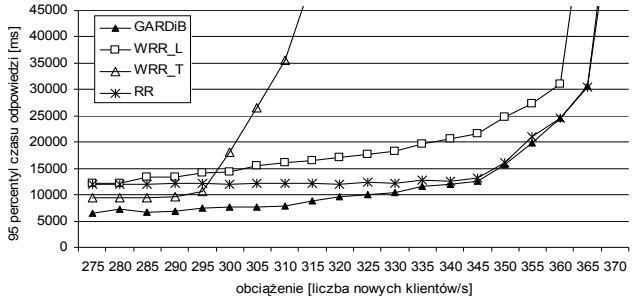


d)

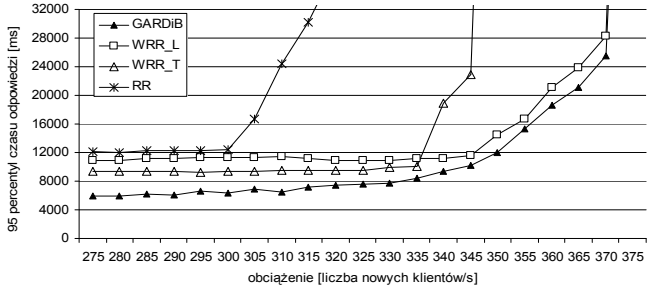


Rys. 3.13. Średnie czasy odpowiedzi na żądania w funkcji liczby nowych klientów dla wariantów: a) NL/3/25%; USA/3/25%; AU/3/,25%; pośrednik/25%, b) NL/3/30%; USA/3/25%; AU/3/20%; pośrednik/25%, c) NL/1/10%; USA/2/20%; AU/3/30%; pośrednik/40%, d) NL/3/16,6%;USA/3/16,6%; AU/3/16,6%; BR/3/16,6%; PL/3/16,6%; pośrednik/16,6%

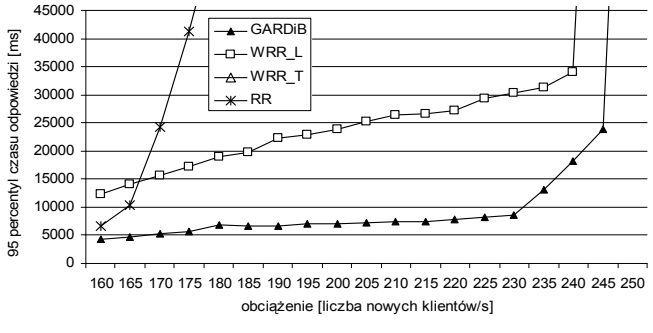
a)



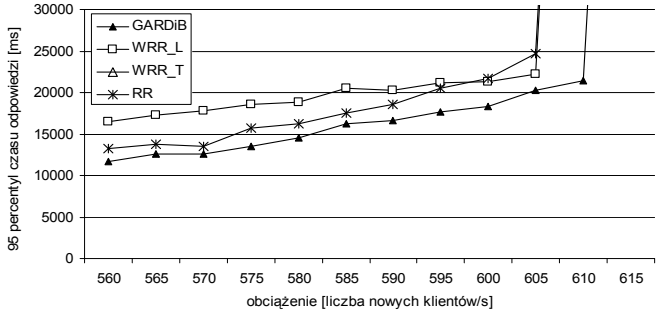
b)



c)



d)



Rys. 3.14. 95 percentyl czasu odpowiedzi strony w funkcji liczby nowych klientów dla konfiguracji: a) NL/3/25%;USA/3/25%; AU/3/,25%; pośrednik/25%, b) NL/3/30%; USA/3/25%; AU/3/20%; pośrednik/25%, c) NL/1/10%; USA/2/20%; AU/3/30%; pośrednik/40%, d) NL/3/16,6%; USA/3/16,6%; AU/3/16,6%; BR/3/16,6%; PL/3/16,6%; pośrednik/16,6%

Z rys. 3.13 oraz 3.14 wynika, że dla metody GARDiB uzyskane zostały najkrótsze średnie czasy odpowiedzi na żądania HTTP oraz najmniejsze wartości 95 percentyla czasu odpowiedzi strony. Dobre wyniki dla tej metody uzyskane zostały w badaniach, w których wykorzystywane były zarówno hetero- jak i homogeniczne serwisy lokalne. Najlepsze wyniki uzyskane zostały, gdy poszczególne serwisy różniły się mocą obliczeniową i były nierównomiernie obciążone. W takim przypadku system webowy pracujący zgodnie z algorytmem GARDiB uzyskał czasy ponadtrzykrotnie krótsze niż dla pozostałych algorytmów. W spotykanych w Internecie systemach webowych bardzo często wykorzystuje się serwisy lokalne różniące się znacznie mocą obliczeniową i wielkością, również rzadko kiedy serwisy lokalne są obciążone równomiernie.

Algorytmem, który również umożliwiał dość dobrą dystrybucje żądania HTTP był algorytm WRR_L, pod kontrolą którego serwis mógł przyjąć we wszystkich badaniach dość duże obciążenie. Jednakże dla tego algorytmu uzyskiwane były długie czasy odpowiedzi dla stron, często nieakceptowalne przez użytkowników.

Kolejne badania dotyczyły czasów podjęcia decyzji dla poszczególnych algorytmów. Badania zostały przeprowadzone w podobny sposób i dla tego samego serwera, jak zostało to opisane w rozdz. 3.1.3. W tabeli 3.4 zaprezentowane zostały średnie czasy podjęcia decyzji oraz średnia liczba decyzji podejmowanych na sekundę.

Tabela 3.4

Liczba decyzji podjętych w ciągu sekundy i średni czas podjęcia decyzji dla różnych algorytmów dystrybucji żądań

Algorytm	**Średnia liczba decyzji podjętych w ciągu sekundy**	**Średni czas podjęcia decyzji**
GARDiB	465 tys.	0,00215 ms
WRR_L	12 mln	0,000066 ms
WRR_T	12 mln	0,000066 ms
RR	20 mln	0,000050 ms

Podobnie jak w przypadku algorytmu LFNRD, czasy podjęcia decyzji dla algorytmu GARDiB są najdłuższe, ale są wystarczająco krótkie, aby nie wpływały negatywnie na czasy odpowiedzi na żądania klientów (czas podjęcia decyzji jest o cztery rzędy mniejszy niż czas odpowiedzi na żądanie). Wydajność pośrednika pracującego pod kontrolą algorytmu GARDiB również jest wystarczająca, aby obsługiwać duże serwisy webowe. Pesymistyczna czasowa złożoność obliczeniowa dla algorytmu podejmowania decyzji w systemie GARDiB wynosi $O(\log N + S)$ (podobnie jak w systemie LFNRD), gdzie N jest liczbą obiektów, które można zażądać. Serwer pośredniczący nawet przy większym obciążeniu nie powinien stać się „wąskim gardłem" systemu webowego. W przypadku, gdyby jednak wydajność serwera pośredniczącego GARDiB była niewystarczająca, można wykorzystać dwa lub więcej serwery umiejscowione w jednej lokalizacji realizujące te same zadania.

Podsumowując można stwierdzić, że otrzymane wyniki badań oraz wyniki opublikowane w [33, 34, 36, 37, 39, 216] wskazują, że algorytm i metoda GARDiB mogłyby być z powodzeniem stosowane w dużych globalnie rozproszonych serwisach webowych, podnosząc znacząco jakość obsługi.

3.3. SYSTEM GARD DYSTRYBUCJI ŻĄDAŃ HTTP W ŚRODOWISKU GLOBALNIE ROZMIESZCZONYCH SERWERÓW WEBOWYCH BEZ SERWERÓW POŚREDNICZĄCYCH

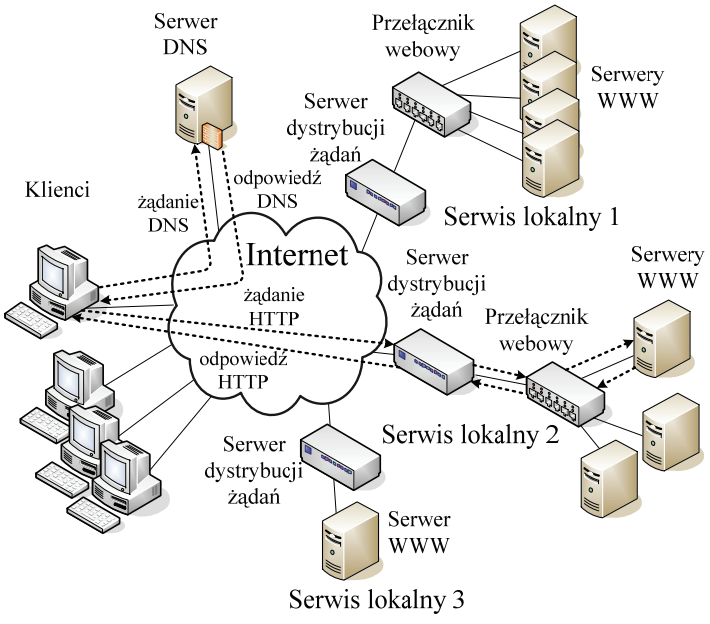
Opisany w rozdz. 3.2 system GARDiB globalnej dystrybucji żądań HTTP, w którym wykorzystywane są serwery pośredniczące, ma wiele zastosowań oraz zalet. Bardzo ważną jego cechą jest to, że system daje dużą kontrolę nad ruchem żądań HTTP nie tylko na poziomie klienta jak przy dotychczas stosowanych systemach, gdzie klient jest przypisywany ze swoim całym ruchem do konkretnego ośrodka w Internecie, ale na poziomie poszczególnych żądań, co umożliwia np. pobranie przez klienta strony, której elementy pochodzą z różnych serwisów lokalnych. Dzięki tak dużej kontroli nad ruchem poszczególne serwisy lokalne nie są przeciążane, a żądania dotyczące obiektów statycznych, których obsługa nie obciąża znacząco procesorów serwerów WWW i serwerów bazodanowych, jest realizowana w najbliższych (w sensie sieciowym) dla klienta serwisach lokalnych.

Jednak budowa systemu GARDiB może być kosztowna, ponieważ wymagane jest utworzenie nie tylko ośrodków dla serwisów lokalnych, ale również uruchomienie w wielu miejscach w Internecie serwerów pośredniczących. Oprócz tego, klienci znajdujący się w dużej odległości od najbliższych pośredników mogą doświadczyć dłuższych czasów odpowiedzi, niż gdyby przyjąć inną metodę dystrybucji żądań. Dlatego też w rozdziale tym opisany zostanie system webowy, pracujący w środowisku globalnie rozmieszczonych serwerów webowych bez serwerów pośredniczących. Prezentowany system nie umożliwia zarządzania żądaniami HTTP z tak dużą kontrolą, jak w przypadku metody GARDiB, ale jego zastosowanie może być bardziej uzasadnione z ekonomicznego punktu widzenia.

3.3.1. OPIS SYSTEMU GARD

System GARD przeznaczony jest do pracy w rozległych sieciach komputerowych, obsługujących dużą liczbę klientów webowych rozmieszczonych w różnych rejonach świata. Na rys. 3.15 zaprezentowany został schemat systemu GARD wraz z zaznaczoną przykładową drogą żądań HTTP. Elementami wchodzącymi w skład systemu GARD są: klienci, serwisy lokalne, serwery dys-

trybucji żądań oraz serwery systemu DNS. Sposób działania i funkcjonalności poszczególnych elementów w systemie determinuje metoda GARD opracowana dla systemu GARD.



Rys. 3.15. Schemat globalnej dystrybucji żądań HTTP z serwerami dystrybucji żądań

Serwis lokalny, podobnie jak w metodzie GARDiB, jest to klaster serwerów webowych, w skład których mogą wchodzić: przełącznik webowy, jeden lub kilka serwerów WWW oraz serwery zaplecza takie jak serwery aplikacji i bazodanowe. Serwisem lokalnym może być również pojedynczy serwer webowy.

Każdy z serwisów lokalnych w systemie może obsłużyć każde z żądań HTTP należące do zbioru dopuszczalnych dla danego systemu webowego żądań.

Serwery dystrybucji żądań umieszczone są w pobliżu serwisów lokalnych i połączone z nimi siecią lokalną. Na każdy pojedynczy serwis lokalny przypada dokładnie jeden serwer dystrybucji żądań. Każdy z serwerów dystrybucji żądań posiada swój adres IP. Serwery te pośredniczą w interakcjach między klientami a serwisami lokalnymi.

Celem działania systemu GARD, podobnie jak systemu GARDiB, jest minimalizowanie czasu odpowiedzi na każde pojedyncze żądanie HTTP. W systemie tym klienci na początku interakcji otrzymują od odpowiedniego serwera DNS adres IP, odpowiadający mnemonicznej nazwie serwisu. Adres ten wskazuje na serwis lokalny, znajdujący się w najbliższej odległości od klienta (podobnie jak w przypadku metody GARDiB). W kolejnych krokach klient wysyła żądania HTTP, które są przekazywane przez serwer dystrybucji żądań bezpo-

średnio do serwisu lokalnego, odpowiadającego danemu serwerowi dystrybucji żądań. Odpowiedzi na żądania HTTP przekazywane są z serwisów lokalnych do serwerów dystrybucji żądań. Jeśli w odpowiedzi przekazywany jest obiekt HTML (szkielet strony), to serwer dystrybucji żądań przekształca dany obiekt w ten sposób, aby adresy obiektów zagnieżdżonych w stronie lub też innych hiperłączy na stronie wskazywały na te same obiekty, ale umiejscowione w tych serwisach lokalnych, dla których czas odpowiedzi na żądania dotyczące danych obiektów były najmniejsze. Przy podejmowaniu decyzji dotyczących alokacji przyszłych żądań brane jest pod uwagę aktualne obciążenie serwisów lokalnych oraz stan sieci na kierunku od klienta do serwisów lokalnych. Odpowiedź zawierająca zmodyfikowany obiekt przekazywana jest bezpośrednio do klienta. W celu ustalenia stanu sieci na kierunku od wszystkich serwisów lokalnych do klienta, przy pobieraniu pierwszego dokumentu HTML adresy obiektów zagnieżdżonych przekształcane są tak, aby z każdego serwisu lokalnego pobrany został co najmniej jeden obiekt. W trakcie nawiązywania połączenia klienta z serwisem lokalnym określany jest stan sieci. Serwery dystrybucji żądań adaptacyjnie modyfikują swoje parametry i dostosowują się do zmieniających warunków pracy w celu polepszenia jakości szacowania czasów odpowiedzi na żądania HTTP.

Poszczególne serwery dystrybucji żądań wymieniają się informacjami o obciążeniu serwisów lokalnych, jak również wiedzą niezbędną do oszacowania czasów obsługi żądań.

Zdefiniujmy pojęcie czasu odpowiedzi w systemie GARD.

Definicja 3.3. Czas odpowiedzi na żądanie HTTP w systemie GARD jest to czas liczony od momentu rozpoczęcia wysłania przez klienta żądania HTTP do momentu otrzymania przez klienta w całości odpowiedzi.

W celu sformułowania zadania projektowego oraz wprowadzenia precyzyjnego opisu działania systemu GARD przyjmuje się następujące dodatkowe oznaczenia:

- y_i – odpowiedź serwera webowego na żądanie x_i, $y_i = \langle h_i, o_i \rangle$ i $y_i \in Y$, gdzie Y jest zbiorem odpowiedzi na żądania HTTP należące do zbioru X,
- h_i – element odpowiedzi y_i, nagłówek odpowiedzi HTTP, jest zbiorem informacji i poleceń, $h_i \in H$, gdzie H jest zbiorem prawidłowo skonstruowanych nagłówków HTTP,
- o_i – element odpowiedzi y_i, żądany w x_i obiekt HTTP, $o_i \in ZO$, gdzie ZO jest zbiorem obiektów dostępnych w serwisie,
- ou_i – wektor adresów obiektów zagnieżdżonych lub hiperłączy zawartych w obiekcie o_i wskazujących na obiekty HTTP udostępniane przez system webowy, $ou_i = \left[u_{i1},...,u_{ig},...,u_{iG_i}\right]$, gdzie G_i jest liczbą obiektów zagnieżdżonych w obiekcie o_i, g jest indeksem adresu, $g = 1,...,G_i$,

u_{ig} – adres obiektu zagnieżdżonego w obiekcie o_i lub hiperłącza do innego obiektu udostępnianego w serwisie webowym,

u'_{ig} – adres obiektu wskazujący ten sam obiekt, co w u_{ig} w innym serwisie lokalnym niż w oryginalnym adresie,

ou'_i – wektor zmodyfikowanych adresów, $ou_i = \left[u'_{i1},...,u'_{ig},...,u'_{iG_i}\right]$,

o'_i – zmodyfikowany obiekt o_i, zawierający adresy wskazane w wektorze ou'_i,

h'_i – zmodyfikowany nagłówek odpowiedzi h_i uzupełniony o żądanie ustawienia w *cookie* dwójki $\left\langle so_i, \bar{t}_i^{so_i j} \right\rangle$, $h'_i = h_i \cup \left\{\left\langle so_i, \bar{t}_i^{so_i j} \right\rangle\right\}$,

y'_i – zmodyfikowana odpowiedź y_i na i-te żądanie, $y'_i = \left\langle h'_i, o'_i \right\rangle$,

$\hat{t}_{ig}^{s}$ – szacowany czas odpowiedzi dla s-tego serwera na żądanie o obiekt o adresie u'_{ig},

$\tilde{t}_{ig}$ – czas odpowiedzi na żądanie o obiekt o adresie u'_{ig},

O_i^{sj} – obciążenie s-tego serwisu lokalnego, widziane z punktu widzenia j-tego klienta w momencie nadejścia i-tego żądania,

$\bar{t}_i^{sj}$ – obciążenie sieci Internet na kierunku od klienta do s-tego serwisu lokalnego, czas RTT zmierzony w trakcie nawiązywania połączenia TCP między j-tym klientem, a s-tym serwisem lokalnym,

j – indeks klienta $j = 1,...,J$, gdzie J jest liczbą klientów,

so_i – numer serwisu lokalnego, przy którym pracuje serwer dystrybucji żądań obsługujący i-te żądanie.

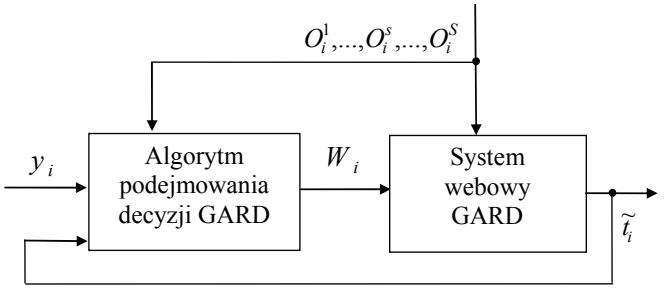
Najważniejszym elementem systemu GARD, w którym podejmowane są decyzje, jest serwer dystrybucji żądań. Poniżej przedstawione zostało zadanie, którego rozwiązaniem jest projekt serwera dystrybucji żądań.

Zadanie projektowe: należy opracować projekt serwera dystrybucji żądań, który na podstawie wiedzy $U_i^1,...,U_i^S$ oraz obciążeń $O_i^{1j},...,O_i^{Sj}$ serwisów lokalnych, dla każdej odpowiedzi y_i na żądanie x_i, $i = 1,2,...$, zawierające adresy, które można przedstawić w postaci wektora adresów $ou_i = \left[u_{i1},...,u_{iG_i}\right]$, wskazujących na obiekty HTTP zagnieżdżone w stronie lub hiperłączy do obiektów HTTP wyznaczy wektor decyzji $W_i = \left[w_{i1},...,w_{iG_i}\right]$, wskazujących na serwisy lokalne, które obsłużą żądania dotyczące obiektów wektora ou_i w taki sposób, aby minimalizować czasy odpowiedzi $\tilde{t}_{i1},...,\tilde{t}_{iG_i}$ na żądania dotyczące wskazanych obiektów oraz zmodyfikuje żądanie y_i w y'_i tak, aby nowa odpo-

wiedź y_i' ujmowała wyniki podjętych decyzji. Dodatkowo serwer dystrybucji żądań powinien adaptacyjnie aktualizować swoją wiedzę o systemie $U_i^s\Big|_{s=so_i}$ na podstawie przeszłych czasów odpowiedzi $\tilde{t}_1,...,\tilde{t}_{(i-1)}$ na żądania.

Na rysunku 3.16 przedstawiony został ogólny schemat procesu podejmowania decyzji z udziałem serwera dystrybucji żądań.

Uogólniając, serwer dystrybucji żądań powinien zmodyfikować treści pobieranych przez klienta stron HTML w taki sposób, aby adresy obiektów zagnieżdżonych oraz hiperłączy wskazywały na te serwisy lokalne, które mogą w najkrótszym czasie dostarczyć do klienta żądane treści. Równocześnie powinien aktualizować swoją wiedzę o systemie na podstawie zmierzonych czasów odpowiedzi na żądania.



Rys. 3.16. Schemat prezentujący proces podejmowania decyzji z udziałem serwera dystrybucji żądań

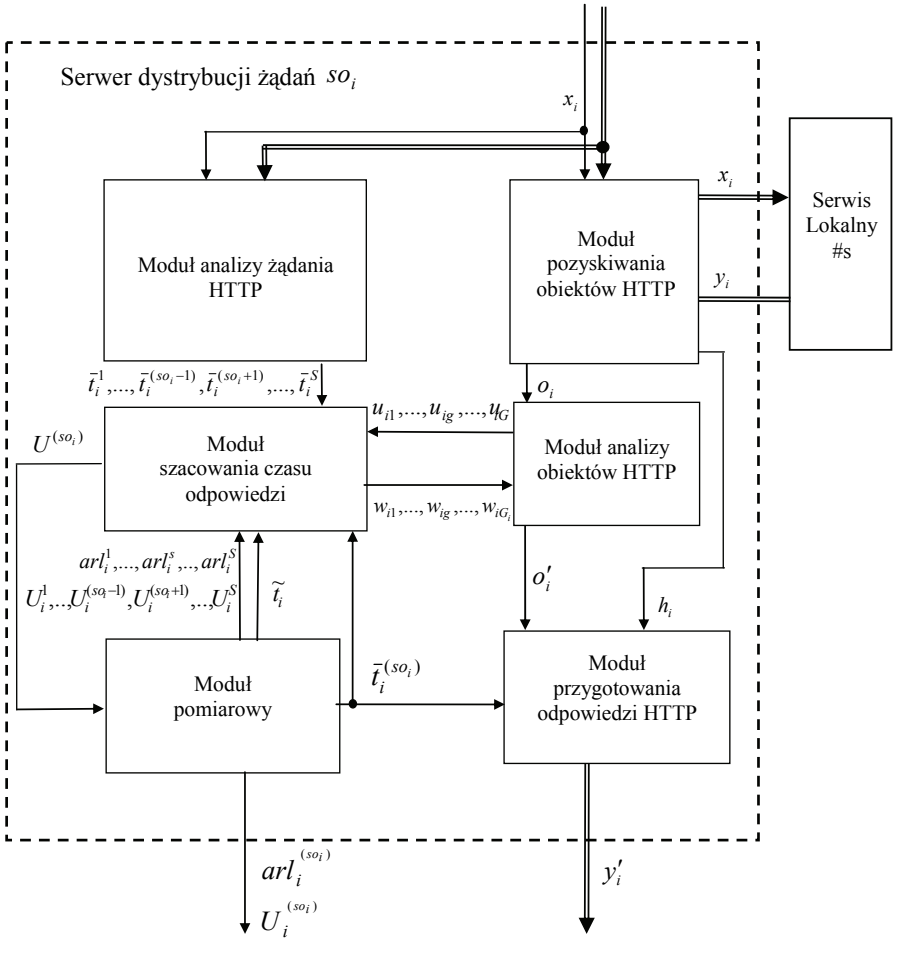
3.3.2. SERWER DYSTRYBUCJI ŻĄDAŃ

W skład serwera dystrybucji żądań wchodzą następujące moduły funkcjonalne: moduł analizy żądania HTTP, moduł pozyskiwania obiektów HTTP, moduł szacowania czasów odpowiedzi, moduł analizy obiektów HTTP, moduł pomiarowy, moduł przygotowania odpowiedzi HTTP. Schemat funkcjonalny serwera dystrybucji żądań przedstawiony jest na rys. 3.17.

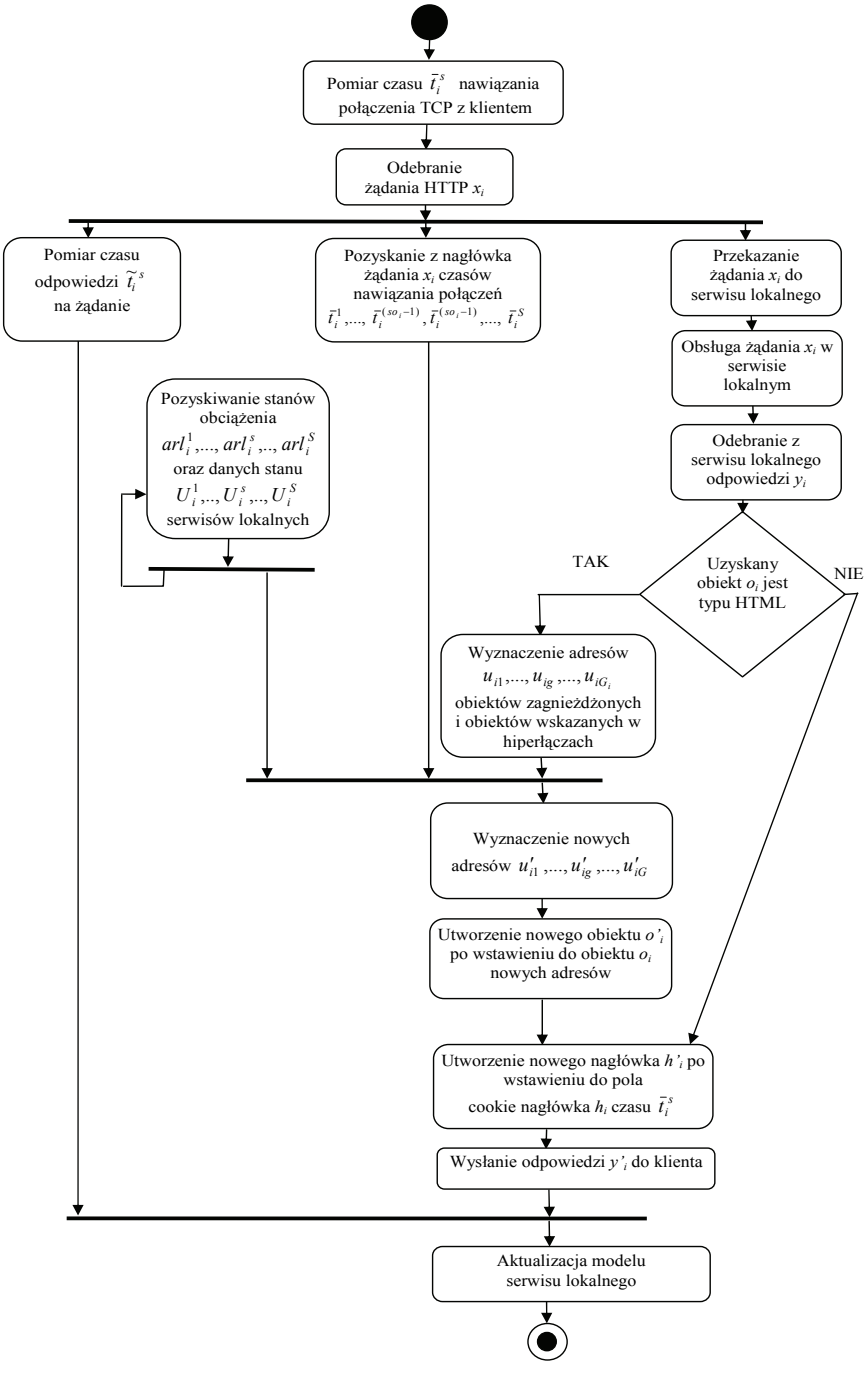
Na rys. 3.18 zaprezentowany został diagram czynności obrazujący działanie serwera dystrybucji żądań. Po nawiązaniu połączenia klient wysyła do serwera dystrybucji żądań żądanie HTTP $x_i = \langle u_i, ck_i \rangle$. Żądanie zawiera dwa istotne elementy: adres u_i żądanego obiektu oraz informacje zawarte w polu *cookie* ck_i. W polu tym zawarte mogą być różne informacje istotne z punktu widzenia działania serwisu, przyjmiemy jednak, że przekazywane będą czasy $\bar{t}_i^{sj}$

nawiązania połączenia klienta z serwisami lokalnymi wraz z numerami serwisów, do których czasy te przynależą $ck_i = \left\{ \left\langle s, \bar{t}_i^{sj} \right\rangle : s \in \{1,...,S\} \right\}$. Żądanie klienta przekazywane jest do modułu analizy żądania HTTP oraz modułu pozyskiwania obiektów HTTP.

Moduł analizy żądania HTTP wyodrębnia z żądania x_i czasy $\bar{t}_i^{1j},...,\bar{t}_i^{(so_i-1)j},\bar{t}_i^{(so_i+1)j},...,\bar{t}_i^{Sj}$ nawiązania połączeń między klientem a wszystkimi serwisami lokalnymi poza serwisem lokalnym obsługującym i-te żądanie. Wartość czasu $\bar{t}_i^{(so_i)j}$ pozyskiwany jest z modułu pomiarowego.



Rys. 3.17. Schemat serwera dystrybucji żądań



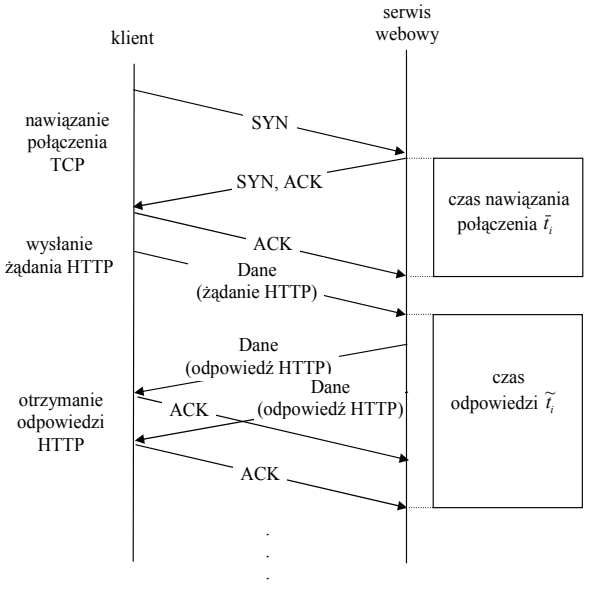
Rys. 3.18. Diagram czynności prezentujący obsługę żądania HTTP przez serwer dystrybucji żądań

Moduł pozyskiwania obiektów HTTP odpowiada za wysłanie żądania HTTP x_i w takiej postaci, w jakiej serwer dystrybucji żądań otrzymał od klienta. Serwis lokalny w odpowiedzi na żądanie klienta wysyła odpowiedź $y_i = \langle h_i, o_i \rangle$. Moduł pozyskiwania obiektów HTTP przekazuje uzyskany obiekt o_i do modułu analizy obiektów HTTP, natomiast nagłówek h_i odpowiedzi do modułu przygotowania odpowiedzi HTTP.

Moduł analizy obiektów HTTP analizuje zawartość otrzymanego obiektu HTTP o_i i wyznacza wektor adresów $ou_i = [u_{i1},...,u_{ig},...,u_{iG_i}]$, gdzie G_i jest liczbą wyodrębnionych z obiektu adresów. Wskazane adresy przekazane zostają do modułu szacowania czasu odpowiedzi. W module szacowania czasu odpowiedzi dla każdego adresu u_{ig}, $g = 1,...,G_i$, podejmowana jest odrębnie decyzja w_{ig}, polegająca na wskazaniu numeru serwisu lokalnego, który powinien obsłużyć żądanie dotyczące danego obiektu. Moduł szacowania czasu odpowiedzi w wyniku swego działania przekazuje do modułu analizy obiektów wektor decyzji W_i. Moduł analizy obiektów przekształca adresy $u_{i1},...,u_{ig},...,u_{iG_i}$ w $u'_{i1},...,u'_{ig},...,u'_{iG}$ w taki sposób, aby nowe adresy wskazywały te same obiekty, w serwisach lokalnych wskazanych w decyzji W_i. Następnie moduł analizy obiektów HTTP tak modyfikuje obiekt o_i w obiekt o'_i, aby ten zawierał zbiór zmodyfikowanych adresów ou'_i. Modyfikacji podlegają jedynie obiekty typu HTML, dla których liczba $G_i > 0$. Jeśli obiekt nie jest typu HTML, to $G_i = 0$, a w konsekwencji $o'_i = o_i$. Dokładniejszy opis działania modułu szacowania czasów odpowiedzi zawarty został w dalszej części tego rozdziału.

Moduł pomiarowy odpowiedzialny jest za uzyskanie przez serwer dystrybucji żądań informacji o stanie obciążenia serwisów lokalnych i stanie obciążenia sieci rozległej. Moduł ten wykonuje również pomiar czasu odpowiedzi $\tilde{t}_i^s$ dla realizowanego i-tego żądania. Czas ten uzyskiwany jest po zakończeniu obsługi żądania i przekazywany jest do modułu szacowania czasu odpowiedzi.

Moduł pomiarowy mierzy również czas $\bar{t}_i^{(so_i)j}$ nawiązania połączenia klienta z serwerem dystrybucji żądań. Czas nawiązania połączenia $\bar{t}_i^{(so_i)j}$ między s-tym serwisem lokalnym a j-tym klientem mierzony jest w momencie nawiązania przez klienta połączenia TCP z serwerem dystrybucji żądań. Jest to czas, jaki upływa od momentu wysłania przez serwer segmentu umożliwiającego nawiązanie połączenia z podniesionymi flagami SYN oraz ACK, do momentu otrzymania przez serwer odpowiedzi klienta potwierdzającej nawiązanie połączenia, czyli otrzymania segmentu z podniesioną flagą ACK. Na rys. 3.19 przedstawiony jest sposób wyznaczania czasu nawiązania połączenia w interakcji klienta z serwerem dla protokołów TCP i HTTP.



Rys. 3.19. Szacowanie czasów nawiązania połączenia i odpowiedzi w interakcji klienta z serwerem dla protokołów TCP i HTTP

Moduł pomiarowy mierzy dodatkowo obciążenie serwisu lokalnego $arl_i^{(so_i)}$. Moduł ten odpowiedzialny jest również za wysłanie do wszystkich serwerów dystrybucji żądań pracujących w serwisie, aktualnych informacji o stanie obciążenia serwisu lokalnego, do którego jest przyporządkowany. Moduł pomiarowy otrzymuje od wszystkich pozostałych serwerów dystrybucji żądań informacje o stanie obciążenia serwisów lokalnych $arl_i^1,...,arl_i^{(so_i-1)},arl_i^{(so_i+1)},...,arl_i^S$. Wraz z wymianą informacji o obciążeniu serwisów lokalnych, moduły pomiarowe poszczególnych serwerów dystrybucji żądań wymieniają się aktualnymi danymi stanów $U_i^1,...,U_i^S$ w ten sposób, że serwer dystrybucji żądań o numerze so_i otrzymuje z pozostałych serwerów dystrybucji żądań informacje $U_i^1,...,U_i^{(so_i-1)},U_i^{(so_i+1)},...,U_i^S$ i wysyła do pozostałych serwerów dystrybucji żądań informacje $U_i^{so_i}$.

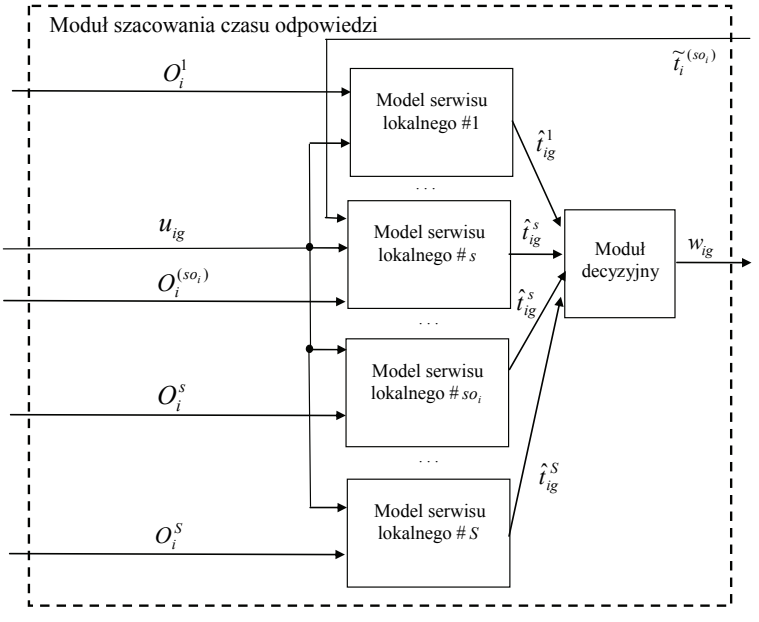
Moduł przygotowania odpowiedzi przygotowuje odpowiedzi HTTP wysyłane do klienta. Odpowiedź HTTP $y_i' = \langle h_i', o_i' \rangle$ posiada zmodyfikowany w module analizy obiektów HTTP obiekt o_i' oraz nagłówek h_i'. Nagłówek odpowiedzi jest odpowiednio modyfikowany w stosunku do oryginalnego tak, że $h_i' = h_i \cup \left\{ \left\langle so_i, \bar{t}_i^{so_i j} \right\rangle \right\}$. Do nagłówka oryginalnego h_i wprowadza się polecanie dla klienta dodania w polu *cookie* wszystkich kolejnych żądań kierowanych do

tego serwisu informacji $\left\langle so_i, \bar{t}_i^{so_i j} \right\rangle$. W ten sposób, w wysyłanych przez klienta żądaniach zamieszczone będą informacje $\bar{t}_i^{1j},...,\bar{t}_i^{sj},...,\bar{t}_i^{Sj}$ niezbędne do oszacowania czasów odpowiedzi na żądania. Nowa odpowiedź y_i' po przygotowaniu jest wysyłana do klienta.

Moduł szacowania czasu odpowiedzi

Zostanie teraz zaprezentowane działanie modułu szacowania czasów odpowiedzi. Rys. 3.20 przedstawia strukturę modułu szacowania czasów odpowiedzi. W skład modułu szacowania czasów odpowiedzi wchodzą: modele serwisów lokalnych i moduł decyzyjny.

Moduł szacowania czasu odpowiedzi posiada S modeli serwisów lokalnych, czyli tyle, ile serwisów lokalnych jest w całym serwisie webowym. Każdy model odpowiada dokładnie jednemu serwisowi lokalnemu.



Rys. 3.20. Schemat modułu szacowania czasu odpowiedzi

Modele serwisów lokalnych umożliwiają szacowanie czasów odpowiedzi $\hat{t}_{ig}^s$, $s = 1,...,S$, $g = 1,...,G$, poszczególnych serwisów lokalnych dla żądań dotyczących obiektów o adresach u_{ig}.

Wejściami do modułu modelu serwisu lokalnego są: adres obiektu HTTP u_{ig} oraz obciążenie O_i^s serwisu lokalnego. Model serwisu lokalnego jest modelem sys-

temu obsługi żądań HTTP opisanym w rozdz. 2. Składowymi obciążenia s -tego serwisu lokalnego O_i^s są: liczba żądań równocześnie obsługiwanych w serwisie lokalnym $a_i^s = arl_i^s$ oraz czas nawiązania połączenia z klientem $b_i^s = \bar{t}_i^{sj}$ [30].

Obciążenie serwisu lokalnego arl_i^s jest pozyskiwane co określony interwał czasowy. Czas nawiązania połączenia, jak zostało to już opisane powyżej, uzyskiwany jest z nagłówka żądania. Jeżeli czas nawiązania połączenia jest nieznany (nie został pozyskany z żądania x_i), to w modelu serwisu lokalnego nie jest wyliczany czas odpowiedzi $\hat{t}_{ig}^s$.

Model serwisu lokalnego adaptacyjnie aktualizuje swoje informacje o serwisie. Adaptacja jest możliwa jedynie dla modelu serwisu lokalnego o numerze so_i, do którego przypisany jest dany serwer dystrybucji żądań. Związane jest to z tym, że jedynie dla tego serwisu lokalnego możliwe jest wykonanie pomiaru rzeczywistego czasu odpowiedzi na żądanie. Modyfikacja parametrów modelu serwisu lokalnego będzie zachodziła za każdym razem, gdy serwer dystrybucji żądań zakończy obsługę żądania. W trakcie modyfikacji parametrów na wejściu modelu serwisu lokalnego podawane będą: obciążenie $O_i^{(so_i)}$, adres żądanego obiektu u_i oraz zmierzony czas odpowiedzi $\tilde{t}_i$ na i -te żądanie.

Ponieważ w serwerze dystrybucji żądań adaptacyjnie aktualizowane są jedynie parametry systemu $U_i^{(so_i)}$ dla modelu serwisu lokalnego so_i, to pozostałe informacje $U_i^1,...,U_i^{(so_i-1)},U_i^{(so_i+1)}...,U_i^S$ przekazywane są do modeli serwerów z modułu pomiarowego zbierającego parametry systemu od pozostałych serwerów dystrybucji żądań. Wymiana informacjami $U_i^1,...,U_i^s,...,U_i^S$ nie została ujęta na rys. 3.20.

Moduł decyzyjny wybiera serwis lokalny, dla którego oszacowany czas odpowiedzi jest najkrótszy. Na wyjściu moduł decyzyjny zwraca decyzję w_{ig}, $w_{ig} \in \{1,...,S\}$, czyli numer serwisu lokalnego, który powinien obsłużyć żądanie. Decyzja w_{ig} wyznaczana jest w następujący sposób:

$$w_{ig} = \begin{cases} s_{\min} : \hat{t}_{ig}^{s_{\min} j} = \min\left\{\hat{t}_{ig}^{sj} : s \in \{1,...,S\}\right\}, \text{gdy} \underset{s\in\{1,...,S\}}{\forall} q_i^{sj} = 1 \\ \min\left\{s : s \in \{1,...,S\} \wedge q_i^{sj} = 0 \wedge s \neq w_{i(g-l)}, l = 1,...,g-1\right\}, \\ \qquad\qquad \text{gdy} \left(\underset{s\in\{1,...,S\}}{\exists} q_i^{sj} = 0\right) \wedge \left(\underset{s\in\{1,...,S\}}{\exists} s \neq w_{i(g-l)}\right) \\ so_i, \text{dla pozostalych przypadków} \end{cases} \tag{3.5}$$

gdzie: $q_i^{sj} = \begin{cases} 1 \text{ gdy znana jest wartość } \bar{t}_i^{sj} \\ 0 \text{ gdy nie jest znana wartosć } \bar{t}_i^{sj} \end{cases}$.

Zgodnie z zależnością (3.5) moduł decyzyjny wybiera ten serwis lokalny, dla którego szacowany czas odpowiedzi jest najkrótszy, o ile dostępne są czasy nawiązania połączeń $\bar{t}_i^{sj}$ dla wszystkich S serwisów lokalnych. Jeśli nie, to wybierane są te serwisy, które nie były jeszcze wybrane w ramach danej strony HTML. Jeśli zostały wybrane już wszystkie serwisy lokalne w ramach danej strony HTML, to wybierany jest serwis lokalny o numerze so_i, do którego przyszło żądanie o i-ty obiekt.

3.3.3. BADANIA DOTYCZĄCE SYSTEMU GARD

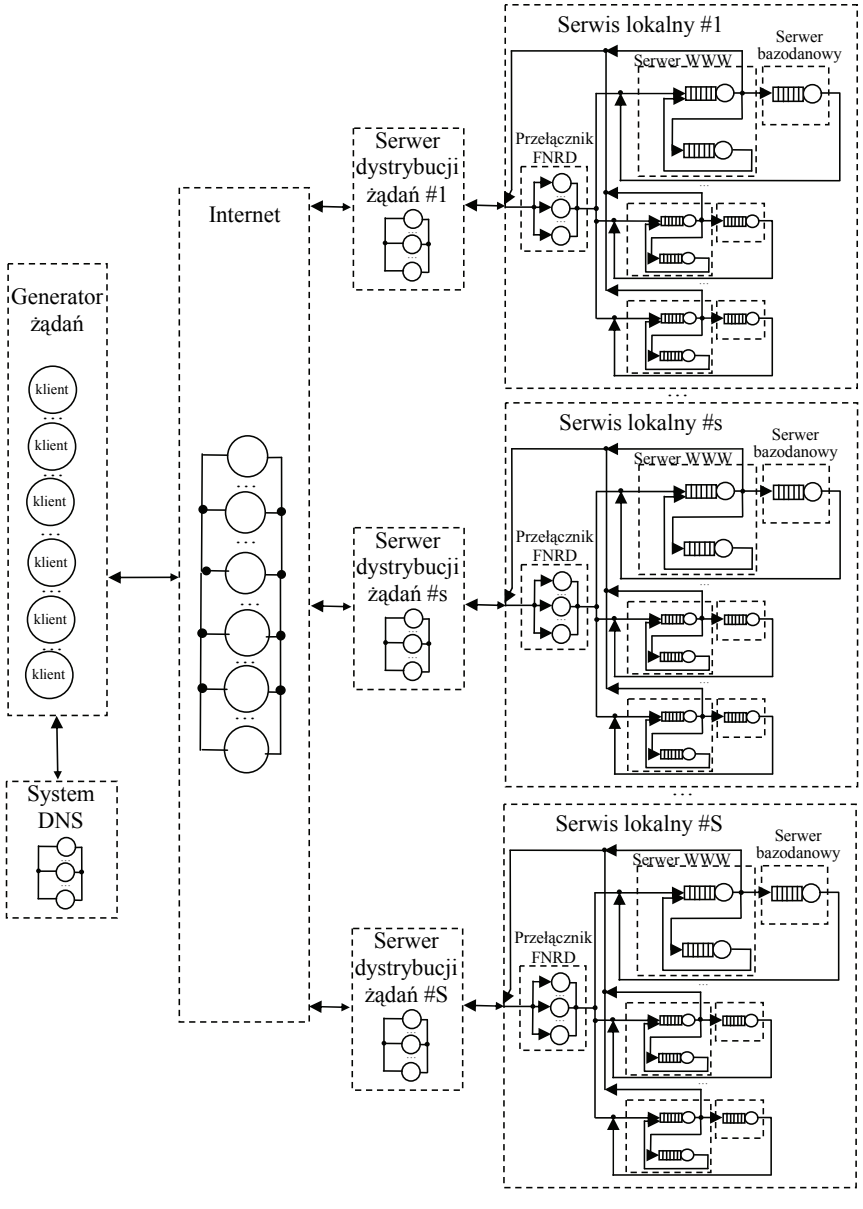
Podobnie jak w przypadku dwóch poprzednich metod dystrybucji żądań, tak i dla metody GARD przeprowadzone zostały badania symulacyjne z wykorzystaniem pakietu CSIM19. W skład programu symulacyjnego wchodziły następujące moduły: moduł generatora żądań, moduł Internetu, moduł DNS, moduł serwera dystrybucji żądań oraz moduł serwisu lokalnego.

Schemat modelu symulacyjnego przedstawiony jest na rys. 3.21. Moduł generatora żądań oraz serwisu lokalnego zostały skonstruowane w ten sam sposób, jak zostało to opisane w rozdz. 3.1.3 oraz rozdz. 3.2.3.

Moduł Internetu został skonstruowany podobnie jak moduł Internetu opisany w rozdz. 3.2.3 z tą jednak różnicą, że do badań nad metodą GARD wykonane zostały dodatkowe badania przepustowości łącza i czasów RTT dla 50 różnych ośrodków webowych w Internecie. W ten sposób uzyskanych zostało 50 różnych scenariuszy, które były wykorzystywane do obliczania czasu przesyłania żądania HTTP i odpowiedzi przez Internet zgodnie z zależnością (3.4). Dla każdego nowo utworzonego przez generator żądań klienta losowanych było tyle scenariuszy, ile serwisów lokalnych pracowało w całym serwisie. Każdy z wylosowanych scenariuszy wykorzystywany był do obliczania czasów przesyłania danych między klientem a konkretnym serwisem lokalnym. Przed każdym eksperymentem symulacyjnym określany był procent użytkowników, dla których czas RTT między klientem a serwisem lokalnym jest najmniejszy.

Moduł DNS wskazywał serwer dystrybucji żądań, do którego klient wyśle żądania HTTP. W module DNS zaimplementowane zostały algorytmy dystrybucji, często stosowane w serwerach DNS [51, 66, 76, 95, 145, 157], są to:

- Weighted Round-Robin Load (WRR_L), dla którego wagi określane były na podstawie liczby żądań obsługiwanych przez poszczególne serwisy lokalne. Moduł DNS otrzymywał informacje o stanie obciążenia poszczególnych serwisów lokalnych co 0,5 s,
- Round Trip Time (RTT), system DNS wykorzystujący ten algorytm wskazywał adres serwisu lokalnego, dla którego w danym momencie czas RTT na linii od klienta do serwisu lokalnego był najkrótszy,
- Round-Robin (RR).



Rys. 3.21. Schemat modelu symulacyjnego w badaniach nad metodą i algorytmem GARD

Przy pracy systemu zgodnie z metodą GARD w module systemu DNS stosowany był algorytm RTT.

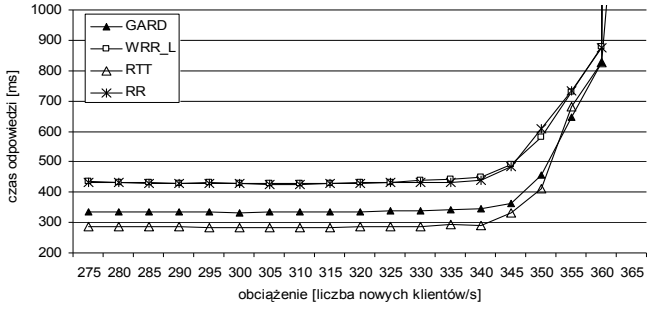
Moduł serwera dystrybucji żądań został tak skonstruowany, aby pracował w sposób opisany w rozdz. 3.3.2. W eksperymentach, w których system webo-

wy nie pracował pod kontrolą algorytmu GARD, moduł serwera dystrybucji żądań przyjmował żądania od klientów i w niezmienionej postaci przekazywał je do serwisów lokalnych, natomiast odpowiedzi z serwisów lokalnych przekazywał bez zmiany do klientów.

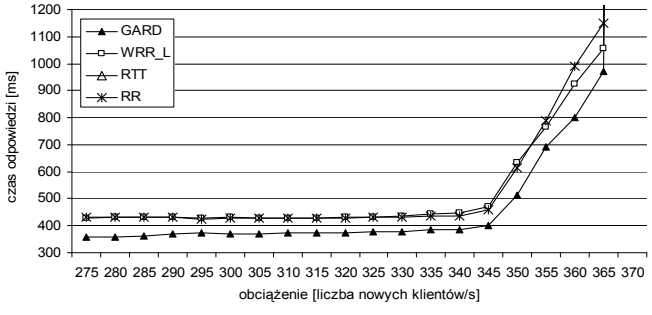
Sposób prowadzenia badań dla metody GARD podobny był do sposobu zastosowanego dla metody GARDiB. Przeprowadzone zostały eksperymenty symulacyjne dla czterech wariantów konfiguracji serwerów. W dwóch pierwszych wariantach wykorzystane zostały trzy serwisy lokalne, z których każdy zawierał jednakowe trzy serwery WWW i bazodanowe. W pierwszym wariancie taka sama liczba klientów (po 33,333%) miała najkrótsze czasy RTT do poszczególnych serwisów lokalnych. Wariant ten został oznaczony 3/33,333%; 3/33,333%; 3/33,333%. W drugim wariancie różna liczba klientów 23,333%, 33,333% oraz 43,333% miała najmniejsze czasy RTT do poszczególnych serwisów lokalnych. Wariant drugi został oznaczony: 3/23,333%; 3/33,333%; 3/43,333%. W wariancie trzecim podobnie jak w poprzednich dwóch wykorzystane zostały trzy serwisy lokalne, z czego w pierwszym znajdował się tylko jeden serwer WWW oraz bazodanowy i dla tego serwisu 16,666% klientów miało najkrótsze czasy RTT, drugi serwis lokalny zawierał dwa serwery WWW i bazodanowe i 33,333% klientów miało dla tego serwisu najkrótsze czasy RTT, natomiast serwis trzeci posiadał trzy serwery WWW i było do niego przydzielonych 50% klientów (na każdy serwer WWW przypadało 16,666% klientów). Wariant trzeci oznaczony został: 1/16,666%; 2/33,333%; 3/50%. W wariancie czwartym wykorzystanych zostało pięć serwisów lokalnych, z czego każdy zawierał po trzy serwery WWW i bazodanowe. Taka sama liczba klientów (po 20%) miała czasy RTT najmniejsze dla poszczególnych serwisów lokalnych. Eksperyment ten oznaczony został: 3/20%; 3/20%; 3/20%; 3/20%; 3/20%.

Na rys. 3.22 przedstawione zostały wykresy średnich czasów odpowiedzi na żądania w funkcji liczby nowych klientów, natomiast na rys. 3.23 zaprezentowane zostały wykresy 95 percentyla czasu odpowiedzi strony w funkcji liczby nowych klientów w poszczególnych eksperymentach symulacyjnych.

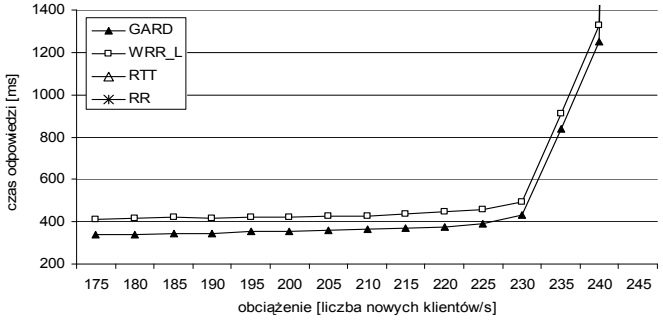
a)



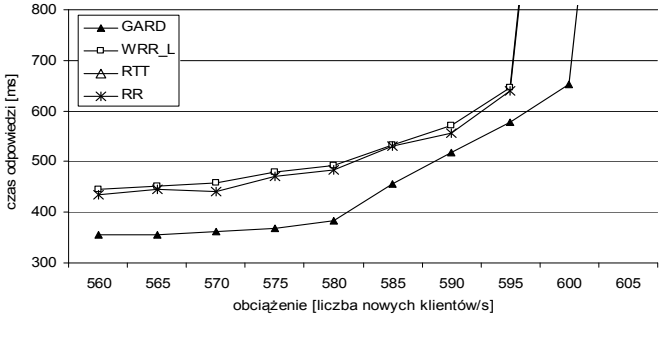
b)



c)

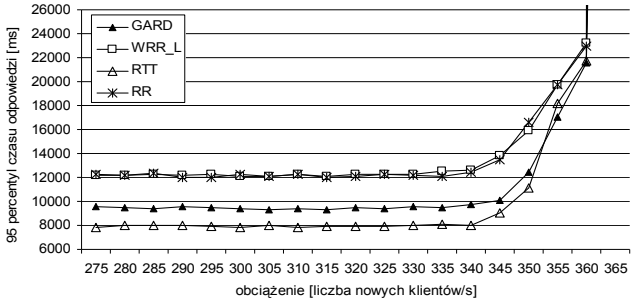


d)

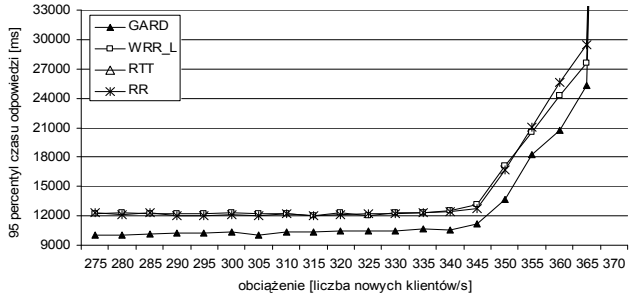


Rys. 3.22. Średnie czasy odpowiedzi na żądanie w funkcji obciążenia dla wariantów: a) 3/33,333%; 3/33,333%; 3/33,333%, b) 3/23,333%; 3/33,333%; 3/43,333%, c) 1/16,666%; 2/33,333%; 3/50%, d) 3/20%; 3/20%; 3/20%; 3/20%; 3/20%

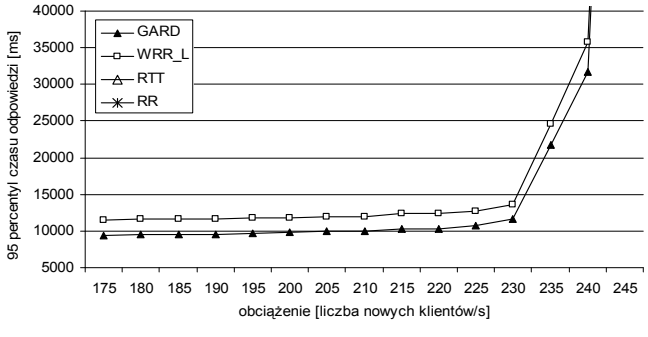
a)



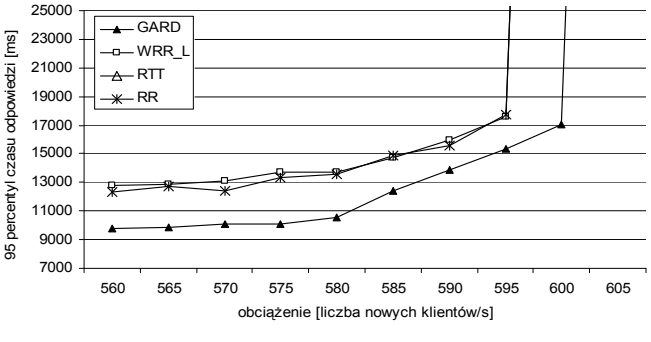
b)



c)



d)



Rys. 3.23. 95 percentyl czasu odpowiedzi strony w funkcji obciążenia dla wariantów: a) 3/33,333%; 3/33,333%; 3/33,333%, b) 3/23,333%; 3/33,333%; 3/43,333%, c) 1/16,666%; 2/33,333%; 3/50%, d) 3/20%; 3/20%; 3/20%; 3/20%; 3/20%

Na podstawie wyników eksperymentów można wnioskować, że system webowy pracujący zgodnie z metodą GARD, skutecznie dystrybuuje żądania, uzyskując przy tym krótkie czasy odpowiedzi na żądania HTTP oraz dla całych stron internetowych. W eksperymencie dla wariantu pierwszego (3/33,333%; 3/33,333%; 3/33,333%,) system pracujący pod kontrolą algorytmu RTT uzyskał najkrótsze czasy odpowiedzi na żądania dla wszystkich obciążeń dlatego, ponieważ nowi klienci byli w miarę równomiernie przydzielani do poszczególnych serwisów lokalnych, nie powodując przeciążenia któregokolwiek z nich, równocześnie czasy przesyłania danych z serwisu do klienta były najkrótsze z możliwych. Natomiast w metodzie GARD każdy klient pobierając pierwszą stronę musiał pobrać obiekty ze wszystkich serwisów lokalnych, również tych znajdujących się w dużej odległości, co powodowało wydłużenie czasów obsługi całych stron. Dla pozostałych wariantów, w których klienci nie byli równomiernie rozmieszczeni lub serwisy lokalne zawierały różną liczbę serwerów WWW, system webowy pracujący zgodnie z metodą GARD uzyskiwał najkrótsze czasy odpowiedzi na żądania. Wynika to z faktu, że z jednej strony wybierane były w trakcie pracy serwisy lokalne znajdujące się najbliżej klientów, z drugiej jednak strony, system nie dopuszczał do przeciążenia któregokolwiek serwisu. Algorytmem, który dość dobrze dystrybuował obciążenie niezależnie od konfiguracji serwerów i obciążenia, był WRR_L, dla którego jednak uzyskiwane były dłuższe czasy obsługi w stosunku do GARD. Na wykresach prezentowanych dla wariantów drugiego, trzeciego i czwartego nie zostały ujęte wyniki dla wszystkich badanych algorytmów. Wynika to z faktu, że dla niektórych algorytmów wartości czasów odpowiedzi na żądania i 95 percentyl czasu odpowiedzi dla stron były bardzo duże (poza przyjętą skalą).

Przeprowadzone zostały również badania, mające na celu określenie wydajności serwera dystrybucji żądań, czasu podjęcia decyzji oraz wyznaczona została pesymistyczna czasowa złożoność obliczeniowa algorytmu podejmowania decyzji. Ponieważ serwer dystrybucji żądań podejmował decyzję o przekierowaniu poszczególnych żądań tylko dla metody GARD, dlatego też badania prowadzone były tylko dla tej metody. W badaniach mierzony był średni czas podjęcia decyzji dla dokumentu HTML oraz pozostałych obiektów. Średni czas podjęcia decyzji wynosił 0,0025 ms, natomiast maksymalna przepustowość wynosiła 400 tys. decyzji na sekundę, co jest wystarczającą wydajnością dla tego typu urządzeń.

Przy obliczaniu pesymistycznej czasowej złożoności obliczeniowej czasu podjęcia decyzji dla każdego z obiektów HTML brane były pod uwagę następujące istotne czasy: czas analizy obiektu HTML, czasy wyznaczenia klas obiektów zagnieżdżonych i wskazanych w hiperłączach, czasy wyznaczenia serwisu lokalnego dla każdego z obiektów oraz czas aktualizacji parametrów przyjętego modelu. Czas analizy obiektu HTML zależy od liczby wyrażeń zawartych w dokumencie, analiza przeprowadzona musi być dla każdego wyrażenia. Przyjmując, że w dokumencie zawartych jest M wyrażeń, złożoność obliczeniowa analizy obiektu wynosi $O(M)$. Czas wyznaczenia klas obiektów wiąże

się z przeszukaniem drzewa binarnego zawierającego dane dotyczące obiektów, dlatego złożoność obliczeniowa wyznaczania klas obiektów wynosi $O(G \cdot \log N)$, gdzie G jest liczbą wyznaczonych adresów obiektów, N jest liczbą obiektów pobieranych w ramach serwisu. Ponieważ decyzja polegająca na wyznaczeniu serwisu lokalnego do obsługi żądania podejmowana jest dla każdego z obiektów oddzielnie, to złożoność obliczeniowa tego procesu wynosi $O(G \cdot S)$, gdzie S jest liczbą serwisów lokalnych. Czas aktualizacji parametrów przyjętego modelu jest stały, dlatego $O(1)$. Podsumowując, pesymistyczna czasowa złożoność obliczeniowa czasu podjęcia decyzji dla każdego z obiektów HTML wynosi $O(M + G \cdot \log N + G \cdot S)$.

Ze względu na wyniki przeprowadzonych badań wydajności serwera dystrybucji żądań oraz analizy pesymistycznej czasowej złożoności obliczeniowej, w której występują zależności liniowe oraz logarytmiczne, można stwierdzić, że serwer dystrybucji żądań nie powinien stać się „wąskim gardłem" systemu GARD. W przypadku, gdyby jednak wydajność serwera dystrybucji żądań była niewystarczająca, możliwe jest wykorzystanie dwóch lub więcej serwerów umiejscowionych w jednej lokalizacji i realizujących te same zadania.

Na podstawie otrzymanych wyników badań prezentowanych w rozdziale oraz innych badań [200, 202, 205, 206, 2013] nad metodą GARD wnioskować można, że zastosowanie tej metody mogłoby podnieść wydajność serwisu, równocześnie podnosząc zauważalnie jakość obsługi użytkowników.

3.4. PODSUMOWANIE

W rozdziale rozpatrywane były zagadnienia podnoszenia jakości usług w systemach webowych. Zaprezentowane zostały trzy metody umożliwiające podnoszenie jakości usług z wykorzystaniem klastrów serwerów webowych, są to: LFNRD, GARDiB oraz GARD.

Autorska metoda LFNRD przeznaczona jest do stosowania w systemach wykorzystujących pojedynczy klaster serwerów webowych umieszczonych w jednej lokalizacji. Metodę GARDiB stosuje się natomiast w systemach z globalnie rozproszonymi klastrami serwerów webowych. W metodzie tej wykorzystuje się serwery pośredniczące. Ostatnia prezentowana w rozdziale autorska metoda GARD również przeznaczona jest dla systemów z globalnie rozproszonymi klastrami serwerów, nie wymaga ona jednak stosowania serwerów pośredniczących.

Opisane metody mogą być wykorzystane zarówno w małych kilkuserwerowych systemach webowych, jak również w dużych systemach wykorzystujących klastry globalnie rozmieszczone.

Dla wszystkich opisanych metod dystrybucji żądań przeprowadzone zostały badania symulacyjne w ujednoliconym układzie badawczym. Metody

i algorytmy porównane zostały z algorytmami referencyjnymi oraz algorytmami najczęściej spotykanymi w zastosowaniach przemysłowych.

Prezentowane wyniki badań symulacyjnych wskazują, że średnie czasy odpowiedzi na żądania HTTP w systemach wykorzystujących opisane w rozdziale metody były prawie we wszystkich przeprowadzonych eksperymentach krótsze niż dla pozostałych metod wykorzystanych w badaniach. Wskazuje to na skuteczność i wysoką jakość pracy systemów, w których opracowane metody zostały przebadane. Na podstawie zaprezentowanych wyników badań wnioskować można, że należy prowadzić dalsze badania nad opracowanymi metodami w środowisku Internetu z udziałem rzeczywistych klientów internetowych, a być może w dalszym kroku wdrożyć je w urządzeniach przemysłowych.

4. SYSTEMY WEBOWE Z ZADANYM CZASEM ODPOWIEDZI

Do niedawna problematyka tworzenia systemów webowych z zadaną jakością usług nie należała do głównego nurtu zagadnień związanych z jakością usług, zwłaszcza jeśli chodzi o gwarantowanie czasu realizacji usług w Internecie. Jest to związane z tym, że usługi w Internecie realizowane są przeważnie z jakością typu best-effort. Tworzenie systemów realizujących usługi na zadanym poziomie jest w znaczny sposób utrudnione poprzez fakt, że Internet jest tworzony przez liczną grupę niezależnych podmiotów, z których tylko nieliczne oferują usługi gwarantowane.

Ze względu na wciąż wzrastające znaczenie usług realizowanych w Internecie oraz fakt, że wiele dziedzin życia jest stopniowo przenoszonych na platformę internetową lub też realizowanych tylko i wyłącznie na tej platformie, wymagane jest utrzymanie jakości na zadanym poziomie przy realizacji określonych usług. Dlatego też w chwili obecnej następuje znaczący wzrost zainteresowania zagadnieniami nie tylko podnoszenia jakości, ale również jej gwarantowania.

W rozdz. 3 omówione zostały zagadnienia projektowania systemów webowych umożliwiających minimalizowanie czasów odpowiedzi na żądania. Z punktu widzenia użytkowników końcowych pożądana jest jak najkrótsza obsługa pojedynczych żądań, co prowadzić może do szybszego pobrania całych stron internetowych. Czasy pobierania stron dostępnych w serwisie webowym mogą być różne, ze względu na zróżnicowane czasy obsługi żądań dotyczących obiektów należących do stron. Dlatego też w tym samym serwisie webowym, przy jednakowych warunkach przetwarzania mogą pojawić się strony, dla których czasy odpowiedzi dla strony będą kilkukrotnie dłuższe niż dla innych stron. W takim wypadku niejednokrotnie można zaobserwować, że strony proste w obsłudze pobierane są szybko, natomiast te złożone, niejednokrotnie kluczowe dla serwisu, pobierane są na tyle długo, że użytkownicy rezygnują z ich oglądania. Najlepszym rozwiązaniem tej sytuacji byłoby zaproponowanie takiego rozwiązania, aby wszystkie strony były pobierane w możliwie krótkim czasie. Bardzo często jednak ze względu na ograniczenia finansowe oraz techniczne, nie jest możliwa realizacja takiego serwisu. Innym rozwiązaniem może być taka obsługa stron w serwisie, aby strony proste były obsługiwane wolniej (w stosunku do czasu standardowej obsługi), w akceptowalnym przez użytkownika czasie, natomiast strony złożone były obsługiwane szybciej, w taki sposób, aby nie został przekroczony dopuszczalny czas odpowiedzi dla strony.

W literaturze zagadnienia nie zostało opisanych wiele rozwiązań umożliwiających gwarantowanie czasów odpowiedzi dla całych stron webowych, do nielicznych należą między innymi [173, 228]. Wynika to z faktu, że systemy takie w swej konstrukcji muszą wykorzystywać odpowiednie modele elementów systemu, umożliwiające szacowanie czasów odpowiedzi na żądania. Trudność polega na opracowaniu modelu, którego zastosowanie w procesie decyzyjnym nie wpłynie znacząco na czas podejmowania decyzji. Jeden z nielicznych tego typu modeli przedstawiony został w rozdz. 2. Zastosowanie wspomnianego modelu otwiera perspektywy opracowania odpowiednich rozwiązań, umożliwiających tworzenie systemów webowych z zadanym czasem odpowiedzi.

W rozdziale tym przedstawione są trzy nowe propozycje systemów webowych. Dotychczas oferowanie usług na zadanym poziomie jakości wiązało się ze sterowaniem dostępem, czyli w praktyce z odrzucaniem żądań klientów, którzy mają mniejsze znaczenie z biznesowego punktu widzenia lub też z dużym różnicowaniem jakości usług dla różnych grup klientów. W rozwiązaniach proponowanych w tym rozdziale, nie są wykorzystywane techniki sterowania dostępem i w jednakowy sposób obsługiwani są wszyscy kliencie zgłaszający się do systemu webowego. Wykorzystywane są natomiast odpowiednie metody szeregowania i dystrybucji żądań. W omawianych rozwiązaniach proponuje się nowe podejście do sposobów przetwarzania w serwisach webowych: serwisy powinny być traktowane jako dostawcy całych stron WWW, a nie – jak dotychczas – pojedynczych obiektów HTTP. Opisywane dalej metody szeregowania i dystrybucji żądań HTTP zostały tak skonstruowane, aby nie dopuszczać do przekroczenia maksymalnego czasu odpowiedzi $t_{\max}$ całych stron WWW lub, o ile czas ten zostanie przekroczony, dążyć do minimalizowania różnic między maksymalnym czasem obsługi $t_{\max}$ a uzyskanym czasem obsługi strony. Przyjmuje się, że powyższe założenia zostaną spełnione, o ile napływ żądań do systemu nie będzie większy od możliwości obsługi żądań przez system webowy.

Zastosowanie omawianych rozwiązań spowoduje, że przy dużym obciążeniu, nie przekraczającym jednak możliwości systemu, czasy obsługi zarówno małych, prostych stron jak i tych złożonych, wymagających pobierania danych z baz danych, będą do siebie zbliżone, a równocześnie nie będą przekraczały wyznaczonego limitu czasowego $t_{\max}$. Nie będzie więc sytuacji, w których przy dużym obciążeniu serwisu jedni użytkownicy będą obsłużeni w krótkim czasie, pozostali natomiast, wymagający złożonych usług, będą oczekiwali bardzo długo.

Zaznaczyć również należy, że wykorzystanie proponowanych metod nie podniesie wydajności serwisu ani nie przesunie granicy maksymalnej liczby obsługiwanych żądań w jednostce czasu przy dużym obciążeniu.

Maksymalny czas odpowiedzi $t_{\max}$ strony powinien być narzucony przez klienta zlecającego usługę prowadzenia serwisu internetowego, czas ten powinien być jednak tak dobrany, aby był możliwym do osiągnięcia czasem odpowiedzi dla każdej ze stron WWW udostępnianej w ramach danego serwisu.

Przyjmuje się, że czas odpowiedzi strony jest czasem liczonym od momentu pojawienia się pierwszego żądania HTTP dotyczącego danej strony, wysłanego przez danego klienta, do momentu otrzymania odpowiedzi HTTP, dotyczącej ostatniego żądania o obiekt strony wysłanego przez tego samego klienta.

Dla każdego z systemów opisanych dalej, czas odpowiedzi strony będzie definiowany odrębnie wskazując dokładnie, w którym momencie rozpocznie się jego pomiar i, w którym się zakończy.

Prezentowane rozwiązania stanowią przekrój rozwiązań niezbędnych do budowy zarówno małych jednoserwerowych serwisów webowych jak i tych rozbudowanych. Na wstępie w rozdz. 4.1 przedstawiona zostanie koncepcja serwisu pracującego zgodnie z metodą WEDF wykorzystującego jeden serwer webowy oraz serwery zaplecza. W rozdz. 4.2 przedstawiona zostanie koncepcja systemu MLF wykorzystującego klaster serwerów rozmieszczonych w jednej lokalizacji. Natomiast w rozdz. 4.3 zaprezentowana zostanie koncepcja systemu GGARDiB globalnej dystrybucji żądań z wykorzystaniem serwerów pośredniczących.

Opisywane dalej metody i algorytmy są algorytmami heurystycznymi, dla których w celu określenia ich własności przeprowadzone zostały badania symulacyjne.

4.1. SYSTEM WEDF SZEREGOWANIA ŻĄDAŃ W ŚRODOWISKU Z JEDNYM SERWEREM WEBOWYM

Jako pierwszy opisany zostanie system WEDF. System ten przeznaczony jest do stosowania w serwisach webowych obsługiwanych przez jeden serwer WWW.

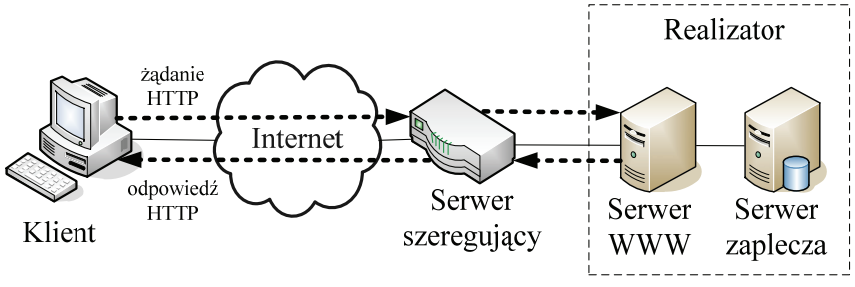
4.1.1. OPIS SYSTEMU WEDF

W systemie WEDF wyróżnić można serwer szeregujący oraz realizator. Sposób działania i funkcjonalności poszczególnych elementów w systemie determinuje metoda WEDF opracowana dla systemu WEDF.

Na rys. 4.1 przedstawiony jest sposób połączenia serwera szeregującego z realizatorem. Serwer szeregujący odbiera żądania HTTP wysyłane od klientów do serwisu, ustawia je w odpowiedniej kolejności, a następnie przesyła do realizatora obsługującego żądania.

Zadaniem serwera szeregującego jest szeregowanie żądań HTTP tak, aby czasy odpowiedzi całych strony WWW były równe lub krótsze od zadanego czasu $t_{\max}$.

Definicja 4.1. Czas odpowiedzi strony WWW w systemie WEDF jest to czas liczony od momentu odebrania przez serwer szeregujący pierwszego żądania HTTP dotyczącego danej strony, wysłanego przez danego klienta, do momentu otrzymania przez serwer szeregujący odpowiedzi HTTP dotyczącej żądania o ostatni obiekt strony wysłanego przez tego samego klienta, przy czym czas odpowiedzi strony pomniejszony jest o czasy, gdy w systemie nie znajdowało się w obsłudze żadne żądanie danego klienta dotyczące danej strony, a występujące między momentem otrzymania przez serwer szeregujący pierwszego i ostatniego żądania dotyczącego danej strony.

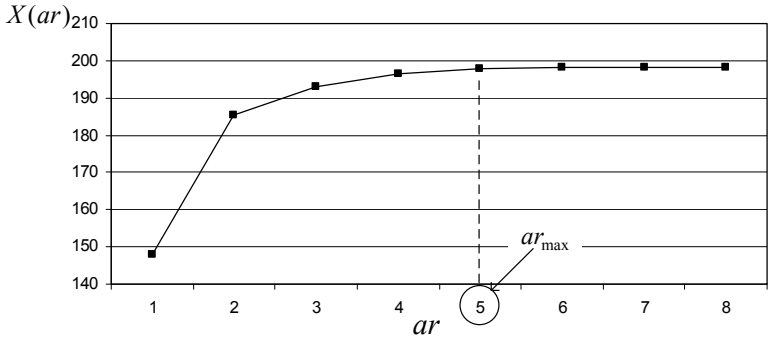


Rys. 4.1. Serwer szeregujący wraz z realizatorem

Serwer szeregujący w opisywanej metodzie może być zarówno niezależnym urządzeniem przesyłającym żądania klientów do serwera WWW jak również może być odpowiednim oprogramowaniem pracującym na komputerze serwera WWW lub też może być częścią oprogramowania samego serwera WWW.

Realizator żądań może składać się z serwera WWW, serwera aplikacji i serwera bazodanowego. Realizator może obsługiwać współbieżnie lub równolegle wiele żądań HTTP. Przyjmuje się, że zastosowanie metody WEDF nie powinno spowodować zmniejszenia wydajności całego serwisu. Przyjmuje się również, że serwer szeregujący będzie dopuszczał do obsługi na realizatorze maksymalnie $ar_{\max}$ żądań równocześnie. Wartość $ar_{\max}$ powinna być tak dobrana, aby być równa minimalnej liczbie żądań, dla której średnia przepustowość systemu $X(ar)$ osiąga wartość maksymalną, gdzie ar jest liczbą żądań obsługiwanych równocześnie na realizatorze, a przepustowość systemu jest liczbą żądań HTTP obsługiwanych w jednostce czasu. Wartość $ar_{\max}$ powinna być liczbą najmniejszą z możliwych do przyjęcia tak, aby w serwerze szeregującym utworzyła się kolejka żądań i równocześnie na tyle dużą, aby serwis WWW osiągnął wydajność zbliżoną do maksymalnej.

Na rys. 4.2 przedstawiony jest przykładowy wykres zależności przepustowości $X(ar)$ od liczby ar równocześnie obsługiwanych żądań wraz z odpowiednio wskazaną liczbą $ar_{\max}$.



Rys. 4.2. Wykres zależności przepustowości od liczby równocześnie obsługiwanych żądań

Zdefiniujmy pojęcie czasu odpowiedzi na pojedyncze żądanie HTTP w systemie WEDF.

Definicja 4.2. Czasem odpowiedzi $\tilde{t}_i$ na żądanie HTTP x_i w systemie WEDF jest czas, który upływa od momentu rozpoczęcia przesyłania żądania HTTP z serwera szeregującego do realizatora, do momentu uzyskania w całości odpowiedzi przez serwer szeregujący.

W skład czasu odpowiedzi na żądanie wchodzi czas transmisji żądania HTTP z serwera szeregującego do serwera WWW, czas przebywania żądania na serwerze WWW, serwerze aplikacji i bazodanowym oraz czas transmisji obiektu HTTP z serwera WWW do serwera szeregującego.

Podstawowym elementem systemu WEDF sterującym pracą systemu jest serwer szeregujący WEDF. W celu opisania działania serwera przyjmuje się następujące dodatkowe oznaczenia:

$t_{\max}$ – zadany nieprzekraczalny czas odpowiedzi dla strony,

i_{p_i} – czas odpowiedzi całej strony o identyfikatorze p_i,

j_i – identyfikator użytkownika, który wysłał i-te żądanie,

p_i – identyfikator strony, w ramach której dany obiekt jest żądany,

d_i – termin, gdy żądanie powinno zostać przesłane z serwera szeregującego do realizatora,

Z_i – wektor klas obiektów należących do strony, a nie pobranych jeszcze w ramach danej strony przez klienta $Z_i = \left[k_i^1,...,k_i^l,...,k_i^L\right]$,

k_i^l – element wektora Z_i, klasa jednego z obiektów nie pobranych jeszcze przez użytkownika w ramach danej strony, $l = 1,...,L$,

L – liczba elementów strony nie pobranych jeszcze przez użytkownika,

tp_{p_i} – czas odpowiedzi strony do momentu nadejścia i-tego żądania,

U_i' – parametry realizatora $U_i' = \left[\hat{t}_{1i}', ..., \hat{t}_{ki}', ..., \hat{t}_{Ki}'\right]$,

$\hat{t}_{ki}'$ – szacowany czas odpowiedzi na żądanie należące do k-tej klasy, $k = 1,..., K$,

K – liczba klas obiektów obsługiwanych w serwisie,

$ar_{\max}$ – maksymalna liczba żądań obsługiwanych na raz przez realizator

$ar(\tau)$ – liczba żądań obsługiwanych równocześnie przez realizator w momencie τ,

$ara(\tau)$ – liczba wszystkich żądań znajdujących się w systemie webowym WEDF w momencie τ,

$\tau_i^{(1)}$ – moment nadejścia i-tego żądania do serwisu (do serwera szeregującego),

$\tau_i^{(2)}$ – moment, gdy i-te żądanie opuszcza kolejkę Q_i,

$ara_{\max}$ – liczba żądań w systemie webowym WEDF, powyżej której następuje załamanie systemu, tj. czasy obsługi poszczególnych żądań wzrastają powyżej akceptowalnych przez klienta wartości,

Q_i – kolejka żądań w serwerze szeregującym,

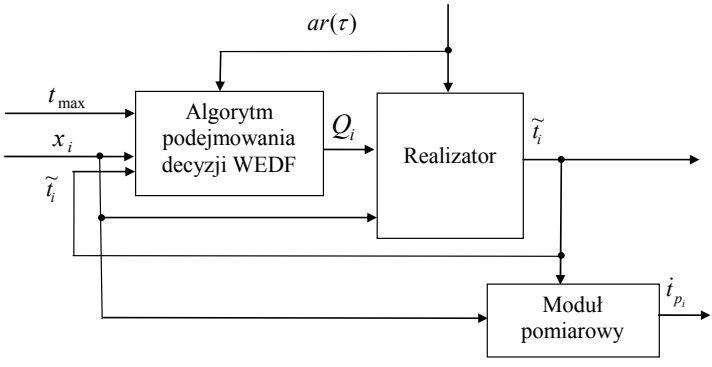
λ – współczynnik równoczesności obsługi żądań.

Zadanie projektowe: należy opracować projekt serwera szeregującego WEDF, który dla każdego nadchodzącego żądania x_i, pod warunkiem $ar(\tau_i^{(1)}) = ar_{\max}$, wyznaczy uszeregowanie dla kolejki żądań HTTP Q_i działającej zgodnie z polityką EDF (ang. Earliest Deadline First), w której wybór kolejności żądań realizowany jest na podstawie przyporządkowanych do żądań x_i terminów d_i, gdzie termin d_i jest tak wyznaczany, aby $\dot{t}_{p_i} \le t_{\max}$, przy założeniach, że $\neg \exists_{p_i}\left[\left(\dot{t}_{p_i} > t_{\max}\right) \wedge \left(\forall_{i=1,2,..} ara(\tau_i^{(1)}) = ar_{\max}\right)\right]$ oraz $\forall_{x_i}\left(ara(\tau_i^{(1)}) < ara_{\max}\right)$. W przypadku, gdy $ar(\tau_i^{(1)}) < ar_{\max}$, żądanie x_i jest przesyłane bezpośrednio do obsługi w realizatorze.

Na rys. 4.3 przedstawiony jest ogólny schemat procesu podejmowania decyzji w systemie WEDF.

Uogólniając, należy zaproponować konstrukcję serwera pośredniczącego, który szereguje żądania w taki sposób, aby czasy odpowiedzi dla stron były krótsze od żądanego czasu $t_{\max}$ przy założeniu, że przy dużym obciążeniu czas $t_{\max}$ jest osiągalny dla wszystkich stron. W celu osiągnięcia założeń w serwerze szeregującym żądania są wstawiane do kolejki Q_i uszeregowanej zgodnie z polityką EDF. Należy tu zaznaczyć, że wyznaczenie kolejki Q_i jest zadaniem

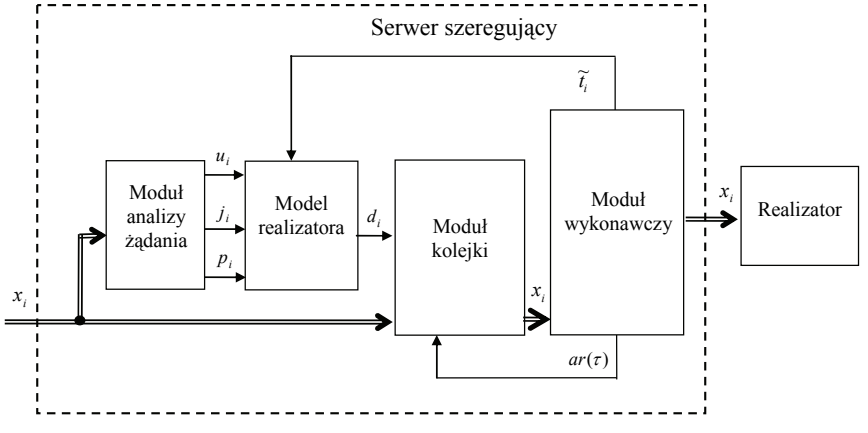
prostym, problemem natomiast jest wyznaczenie terminów d_i, $i = 1,2,...$ dla poszczególnych żądań, zgodnie z którymi żądania zostaną uszeregowane.



Rys. 4.3. Schemat ilustrujący proces podejmowania decyzji w systemie WEDF

4.1.2. SERWER SZEREGUJĄCY WEDF

Serwer szeregujący zawiera moduł analizy żądania, model realizatora, moduł kolejki i moduł wykonawczy. Schemat serwera szeregującego zaprezentowany został na rysunku 4.4.



Rys. 4.4. Schemat funkcjonalny serwera szeregującego

Żądanie, które przychodzi do serwera szeregującego, na wstępie jest analizowane w module analizy żądania. W następnym kroku model realizatora wyznacza termin d_i, gdy powinna rozpocząć się obsługa żądania. Termin ten jest

przekazywany do modułu kolejki, który szereguje żądania zgodnie z wyznaczonymi terminami. Jeżeli liczba żądań obsługiwanych przez realizator jest mniejsza od wyznaczonej eksperymentalnie liczby $ar_{\max}$, żądanie jest pobierane z kolejki i przekazywane do modułu wykonawczego, a następnie dalej do realizatora. Odpowiedź na żądanie jest przekazywana z realizatora do modułu wykonawczego i dalej do klienta.

Zostaną teraz opisane funkcje i działanie poszczególnych modułów serwera szeregującego.

Moduł analizy żądania po odebraniu żądania HTTP $x_i = \langle u_i, ck_i \rangle$, gdzie $ck_i = \{\langle j_i, p_i \rangle\}$, przeprowadza jego analizę i pobiera z niego: adres żądanego obiektu u_i oraz z pola *cookie* identyfikator użytkownika j_i i identyfikator strony p_i**,** w ramach której dany obiekt jest żądany. Adres żądanego obiektu zawsze znajduje się w żądaniu HTTP przekazywanym przez użytkownika do serwisu. Identyfikator użytkownika j_i może być nadany użytkownikowi w momencie pierwszego wejścia do serwisu w ramach sesji i może być przekazany do przeglądarki, a następnie z przeglądarki do serwisu w żądaniu HTTP w informacjach zawartych w polu *cookie* żądania. Identyfikator strony p_i również może być przekazany przeglądarce użytkownika w momencie pobierania szkieletu strony, a następnie podobnie jak powyżej, z przeglądarki do serwisu w żądaniu HTTP w informacjach zawartych w *cookie*.

Model realizatora wyznacza termin d_i dla żądania x_i wskazujący na moment, gdy żądanie powinno zostać przesłane do obsługi przez realizator. Dokładniejszy opis działania tego modułu znajduje się w dalszej części tego punktu.

Moduł kolejki zawiera kolejkę żądań Q_i i jest odpowiedzialny za wstawienie żądania HTTP do kolejki zgodnie z polityką szeregowania żądań. Moduł kolejki otrzymuje z modułu wykonawczego informację o liczbie żądań $ar(\tau_i^{(1)})$ równocześnie obsługiwanych przez realizator w momencie $\tau_i^{(1)}$ nadejścia żądania x_i. Przyjmuje się, że jeżeli liczba żądań równocześnie aktywnie obsługiwanych na realizatorze jest $ar(\tau_i^{(1)}) < ar_{\max}$, wtedy żądanie HTTP x_i nie jest wstawiane do kolejki, lecz jest bezpośrednio przekazywane do modułu wykonawczego i dalej do obsługi w realizatorze. Jeżeli $ar(\tau_i^{(1)}) = ar_{\max}$, to żądanie wstawiane jest do kolejki uszeregowanej zgodnie z terminami d_i, wyznaczonymi przez model realizatora dla każdego żądania wstawianego do kolejki Q_i. Na początek kolejki wstawiane są żądania z najwcześniejszymi terminami, natomiast na koniec z najpóźniejszymi zgodnie z polityką EDF tak, że $Q_i = \left(x_{(i-m_1)}, \ldots, x_{(i-m_g)}, \ldots, x_{(i-m_G)}\right)$, $Q_i \in D$, $i = 1,2,\ldots$, $\underset{g \in \{1,\ldots,G\}}{\forall} m_g \in \{0,\ldots,i-1\}$,

gdzie

$$D = \left\{ \left(x_{(i-m_1)},...,x_{(i-m_g)},...,x_{(i-m_G)}\right): \underset{d_{(i-m_{(g-f)})}, d_{(i-m_g)}, \text{gdzie } g\in\{1,...,G\}, f\geq 0,\, g-f\in\{1,...,G\}}{\forall} d_{(i-m_{g-f})} \leq d_{(i-m_g)} \right\}$$

, $x_{(i-m_g)}$ jest żądaniem umieszczonym na g-tej pozycji kolejki i $x_{(i-m_g)} \in X$, $d_{(i-m_g)}$ jest terminem przyporządkowanym do $x_{(i-m_g)}$, G jest liczbą żądań w kolejce.

Żądanie jest pobierane z kolejki Q_i i przekazywane do modułu wykonawczego tylko wtedy, gdy wszystkie żądania umieszczone w kolejce przed nim opuszczą kolejkę oraz gdy wystąpi zdarzenie w pewnym momencie $\tau_i^{(2)}$ takie, że nastąpi zmniejszenie liczby żądań równocześnie obsługiwanych, co da w rezultacie $ar(\tau_i^{(2)}) < ar_{\max}$.

W omawianym systemie szeregującym nie jest stosowane wywłaszczanie, ponieważ wymagałoby to ingerencji w działanie realizatora.

Moduł wykonawczy wysyła żądanie otrzymane z modułu kolejki do obsługi przez realizator. Obiekt otrzymany w odpowiedzi na żądanie HTTP jest przesyłany przez moduł wykonawczy do klienta. Moduł wykonawczy może równocześnie nadzorować obsługę przez realizator maksymalnie do $ar_{\max}$ żądań.

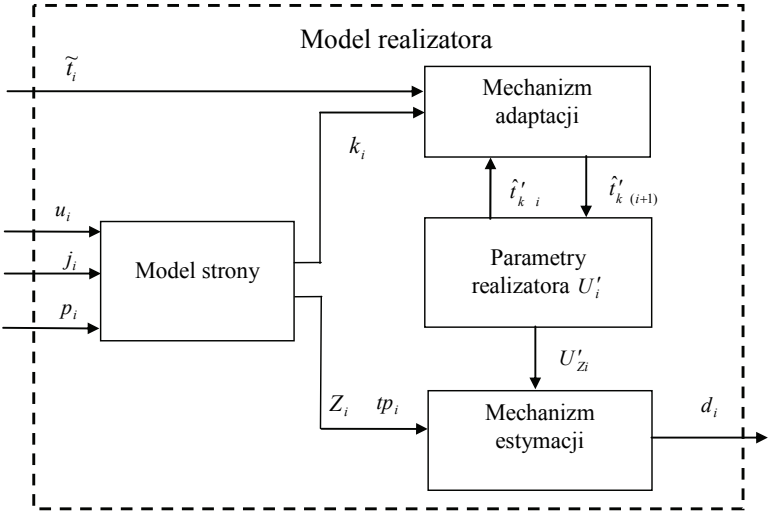
Moduł wykonawczy mierzy czas $\tilde{t}_i$ obsługi żądania x_i i po zakończeniu obsługi przekazuje czas do modelu realizatora.

Model realizatora

Przejdźmy teraz do dokładniejszego opisu modelu realizatora. Model ten wyznacza termin d_i. W module realizatora wyróżnić można 4 podmoduły: model strony, mechanizm estymacji, parametry realizatora i mechanizm adaptacji. Na rys. 4.5 przedstawiony został schemat modelu realizatora.

Model strony przechowuje informacje o obiektach HTTP udostępnianych w serwisie, ich wielkości, rodzaju (czy są dynamiczne czy statyczne) i przynależności poszczególnych obiektów do całych stron. Informacje te mogą się zmieniać w czasie i są aktualizowane w modelu serwisu zwłaszcza w przypadku stron, których szkielet generowany jest dynamicznie. Na wejściu model strony przyjmuje informacje o adresie żądanego obiektu u_i, identyfikatorze użytkownika j_i i identyfikatorze strony p_i. Model strony przechowuje informacje o obiektach pobranych przez użytkownika w ramach pojedynczej strony WWW. Model ten przekazuje również informacje o klasie k_i żądanego obiektu do modułu mechanizmu adaptacyjnego, gdzie $k_i \in \{1,...,K\}$. Klasa obiektu wyznaczana jest w podobny sposób jak zostało to opisane w rozdz. 2.2.

Model strony przekazuje do modułu mechanizmu estymacji wektor $Z_i = \left[k_i^1,...,k_i^l,...,k_i^L\right]$ klas obiektów należących do strony, a nie pobranych jeszcze w ramach danej strony przez klienta, gdzie L jest liczbą obiektów jeszcze nie pobranych. Klasa k_i^1 jest klasą żądanego obiektu (w dalszych oznaczeniach w tym podrozdziale k_i^1 jest również oznaczane jako k_i). Dodatkowo przekazywany jest również czas tp_{p_i} obsługi strony p_i. Czas ten mierzony jest od momentu przyjścia pierwszego żądania, wysłanego przez klienta o obiekt strony do momentu nadejścia żądania x_i, czas ten jest pomniejszony o czasy, w których w serwisie w obserwowanym okresie nie znajdowało się żadne żądanie wysłane przez danego klienta dotyczące danej strony.



Rys. 4.5. Schemat modelu realizatora

Moduł parametrów realizatora przechowuje informacje o czasach odpowiedzi dla obiektów należących do poszczególnych klas. Moduł ten oznaczony został U'_i, przy czym $U'_i = \left[\hat{t}'_{1i},...,\hat{t}'_{ki},...,\hat{t}'_{Ki}\right]$. Parametr $\hat{t}'_{ki}$ jest szacowanym czasem odpowiedzi na żądanie należącego do klasy k, $k = 1,...,K$ przy obciążeniu realizatora $ar_{\max}$.

Moduł parametrów realizatora przekazuje do mechanizmu estymacji czasy odpowiedzi $U'_{Zi} = \left[\hat{t}'_{k^1 i},...,\hat{t}'_{k^l i},...,\hat{t}'_{k^L i}\right]$ na żądania dotyczące obiektów, których klasy wskazane zostały w Z_i.

Mechanizm estymacji wyznacza termin d_i zgodnie z zależnością:

$$d_i = \tau_i^{(1)} + \Delta d_i - \hat{t}'_{k^1 i}. \tag{4.1}$$

Wartość Δd_i jest czasem, jaki może być przeznaczony na oczekiwanie żądania w kolejce i obsługę żądania, Δd_i wyznaczane jest zgodnie z zależnością:

$$\Delta d_i = \hat{t}'_{k^1 i} \frac{(t_{\max} - tp_i)}{\lambda \sum_{l=1}^{L} \hat{t}'_{k^l i}}, \tag{4.2}$$

gdzie λ jest współczynnikiem równoczesności obsługi żądań. Wartość λ zależy od średniej liczby równocześnie obsługiwanych w systemie żądań wysłanych przez jednego użytkownika w trakcie pobierania obiektów należących do tej samej strony. Wartość współczynnika λ powinna zostać dobrana w sposób eksperymentalny.

Przedział czasu Δd_i jest wprost proporcjonalny do czasu $\hat{t}'_{k^1 i}$ obsługi żądania w realizatorze i zależy od pozostałego jeszcze czasu odpowiedzi $t_{\max} - tp_i$ dla całej strony. Wartość Δd_i zależy również od czasu $\lambda \sum_{l=1}^{L} \hat{t}'_{k^l i}$, który należy przeznaczyć na obsługę żądań dotyczących obiektów nie pobranych jeszcze przez użytkownika w ramach danej strony. Może się zdarzyć, że termin d_i będzie wskazywał na moment wcześniejszy niż $\tau_i^{(1)}$ i będzie to zachodziło, gdy $\Delta d_i < \hat{t}'_{k^1 i}$. Może się również zdarzyć sytuacja, że Δd_i przyjmie wartość ujemną, jeżeli czas odpowiedzi dla strony tp_i będzie większy od granicznego czasu $t_{\max}$.

Moduł mechanizmu adaptacji aktualizuje czasy obsługi żądań zawarte w zbiorze danych stanu U'_i. Adaptacja następuje po zakończeniu obsługi żądania x_i należącego do klasy k_i. Aktualizacja zbioru danych stanu następuje jedynie wtedy, gdy przed obsługą żądanie zostanie wstawione do kolejki.

Szacowany czas odpowiedzi $\hat{t}'_{k\ i}$ aktualizowany jest zgodnie ze wzorem

$$\hat{t}'_{k\ (i+1)} = \hat{t}'_{k\ i} + \hat{\eta}(\tilde{t}_i - \hat{t}'_{k\ i}), \tag{4.3}$$

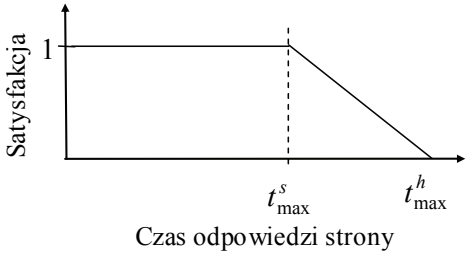
gdzie $\hat{\eta}$ jest współczynnikiem adaptacji, a jego wartość powinna się zawierać w przedziale $(0,1)$. Wstępne badania wykazały, że wartość $\hat{\eta}$ powinna wynosić około 0,1.

4.1.3. OCENA JAKOŚCI PRACY SYSTEMÓW WEBOWYCH Z ZADANYM CZASEM ODPOWIEDZI

Zmiana podejścia do zagadnień przetwarzania żądań HTTP w serwisach webowych oraz wyznaczenie nowych celów wymaga wskazania adekwatnych metod oceny pracy serwisów webowych. Dlatego też do oceny metod WEDF, MLF oraz GGARDiB proponuje się wykorzystanie średniej wartości funkcji zysku nazwanej na potrzeby pracy satysfakcją. Proponowana funkcja zysku jest często wykorzystywana do oceny działania miękkich systemów czasu rzeczywistego [58, 113]. Satysfakcja odzwierciedla poziom zadowolenia użytkownika z czasu odpowiedzi dla całej żądanej strony WWW. Satysfakcja użytkownika zależy od czasu odpowiedzi strony zgodnie z zależnością:

$$Satysfakcja(\dot{t}) = \begin{cases} 1 \quad \text{gdy } \dot{t} \geq 0 \text{ i } \dot{t} < t^s_{\max} \\ 0 \quad \text{gdy } \dot{t} > t^h_{\max} \\ \dfrac{t^h_{\max} - \dot{t}}{t^h_{\max} - t^s_{\max}} \text{ w pozostalych przypadkach} \end{cases} \tag{4.4}$$

gdzie: $\dot{t}$ jest czasem odpowiedzi strony, $t^s_{\max}$ jest czasem, który można przekroczyć, ale należy się starać tego unikać, natomiast $t^h_{\max}$ jest nieprzekraczalnym czasem obsługi, po którym użytkownik rezygnuje z oglądania strony WWW. Czasy $t^s_{\max}$ oraz $t^h_{\max}$ powinny zostać narzucone przez zleceniodawcę zamawiającego usługę obsługi danego serwisu. Przy czym wartość $t^s_{\max}$ powinna być równa wartości czasu $t_{\max}$. Na rys. 4.6 przedstawiona została w graficzny sposób funkcja satysfakcji.

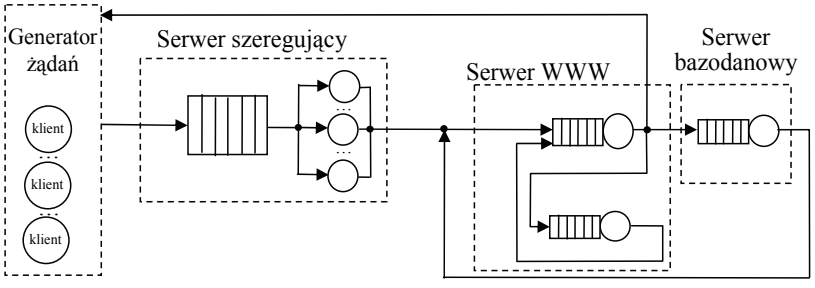


Rys. 4.6. Satysfakcja w funkcji czasu odpowiedzi dla strony

4.1.4. BADANIA DOTYCZĄCE SYSTEMU WEDF

Zostały przeprowadzone badania określające jakość pracy systemu WEDF. Badania miały charakter symulacyjny. W programie symulacyjnym wyróżnić można następujące moduły funkcjonalne: moduł generatora, moduł serwera szeregującego, moduł serwera WWW oraz moduł serwera bazodanowego.

Schemat modelu symulacyjnego przedstawiony jest na rys. 4.7.



Rys. 4.7. Schemat modelu symulacyjnego w badaniach nad metodą WEDF

Moduły serwera WWW i bazodanowego zostały skonstruowane w sposób opisany w rozdz. 3.1.3.

Moduł generatora żądań HTTP został nieznacznie przebudowany w stosunku do modułu generatora opisanego w rozdz. 3.1.3. Obecnie przeglądarki internetowe pobierając stronę WWW, pobierają najpierw szkielet HTML strony, a następnie otwierają kilka połączeń TCP, w ramach których pobierają obiekty zagnieżdżone na stronie. Taki sposób działania przeglądarek ma wpływ na czas pobierania całej strony internetowej i może mieć znaczenie przy algorytmach szeregowania żądań. W dokumencie RFC 2616 [79] zaleca się, aby liczbę równoczesnych połączeń z serwerem WWW w ramach pobierania jednej strony ograniczyć do trzech. Dlatego też moduł generatora żądań został w taki sposób przebudowany, aby najpierw pobierany był obiekt typu HTML, a następnie dla każdego klienta tworzone były dwa dodatkowe wątki pobierające kolejne obiekty danej strony. Każdy z wątków po pobraniu obiektu zatrzymywał swoje działanie na czas namysłu przeglądarki, a następnie kontynuował działanie dopóty, dopóki nie zostaną pobrane wszystkie obiekty danej strony.

W module serwera szeregującego zaimplementowane zostały cztery metody i algorytmy szeregowania żądań:

- WEDF – serwer szeregujący działał zgodnie z opisem metody zawartym w rozdz. 4.1.1,
- FIFO – zgodnie z tą metodą szeregowania żądania HTTP były obsługiwane w kolejności ich nadejścia do obsługi, kolejne żądania były pobierane do obsługi z kolejki, gdy liczba żądań obsługiwanych przez realizator była mniejsza niż $ar_{\max}$,

- Shortest Job First (SJF) – żądania były szeregowane w kolejce zgodnie z algorytmem, przy czym czasy odpowiedzi na żądania wyliczane były w sposób podany w rozdz. 3.1.2,
- Direct Service (DS) – żądania nie były umieszczane w kolejce, lecz bezpośrednio po nadejściu żądanie było przekazywane do obsługi na realizatorze bez względu na to, jaka liczba żądań była aktualnie obsługiwana w realizatorze.

Badania symulacyjne przeprowadzone zostały dla czterech różnych wariantów konfiguracji systemu webowego. W trzech spośród czterech wariantów eksperymenty prowadzone były dla tego samego systemu webowego. Przyjęte zostało, że dla ustalonego czasu $t^s_{\max}$ dobierany był czas $t^h_{\max}$, którego wartość była dwukrotnie większa od wartości $t^s_{\max}$. Prowadzone były symulacje dla wariantów: $t^s_{\max} = 300\,\text{ms}$ i $t^h_{\max} = 600\,\text{ms}$, $t^s_{\max} = 400\,\text{ms}$ i $t^h_{\max} = 800\,\text{ms}$, $t^s_{\max} = 500\,\text{ms}$ i $t^h_{\max} = 1000\,\text{ms}$. W czwartym wariancie czasy obsługi żądań na serwerze bazodanowym zostały wydłużone dwukrotnie, w ten sposób zamodelowany został serwis webowy, w którym występują „lekkie" strony statyczne oraz strony dynamiczne, których obsługa jest dość długa, a co za tym idzie, zwiększyło się zróżnicowanie czasów odpowiedzi dla stron. W czwartym eksperymencie przyjęte zostały następujące czasy: $t^s_{\max} = 500\,\text{ms}$ i $t^h_{\max} = 1000\,\text{ms}$.

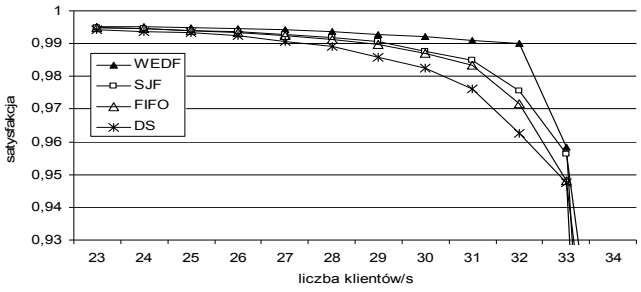
Na rys. 4.8 przedstawione zostały wykresy zależności średniej wartości satysfakcji od obciążenia (liczby nowych klientów generowanych na sekundę). Natomiast na rys 4.9a zaprezentowany został wykres średniej wartości czasu odpowiedzi na żądanie w funkcji obciążenia, a na rys. 4.9b dystrybuanty czasów odpowiedzi dla całych stron przy obciążeniu 32 klientów/s, dla przyjętych czasów $t^s_{\max} = 300\,\text{ms}$ i $t^h_{\max} = 600\,\text{ms}$.

W przeprowadzonych badaniach serwis pracujący pod kontrolą algorytmu szeregowania WEDF uzyskał największe wartości satysfakcji (rys. 4.8), drugi w kolejności był algorytm SJF.

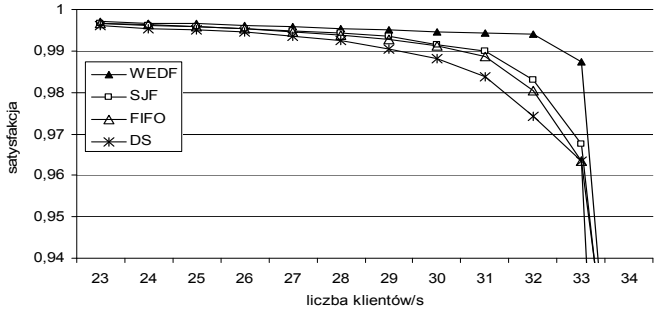
Dla małych obciążeń serwisu dla wszystkich metod szeregowania uzyskane zostały zbliżone wartości satysfakcji. Dla dużych obciążeń, czyli w sytuacjach, gdy użytkownicy długo oczekują na złożone strony dynamiczne, uzyskane zostały wyraźnie lepsze wyniki dla metody WEDF, co oznacza, że czasy odpowiedzi dla stron „lekkich" wydłużały się nie przekraczając jednak czasu $t^s_{\max}$, natomiast czasy odpowiedzi dla stron złożonych najprawdopodobniej nie wydłużały się. Podobne wyniki uzyskane zostały dla wszystkich konfiguracji przyjętych czasów $t^s_{\max}$ i $t^h_{\max}$ oraz dla obu serwisów.

Z rys. 4.9a wynika, że średnie czasy odpowiedzi na pojedyncze żądania HTTP są w większości wypadków dłuższe dla metody WEDF niż dla SJF i FIFO. Wynika stąd, że dążenie do minimalizacji czasów odpowiedzi nie zawsze prowadzi do podniesienia komfortu pracy użytkowników.

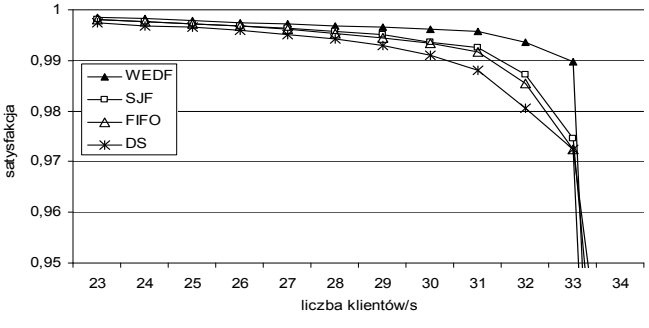
a)



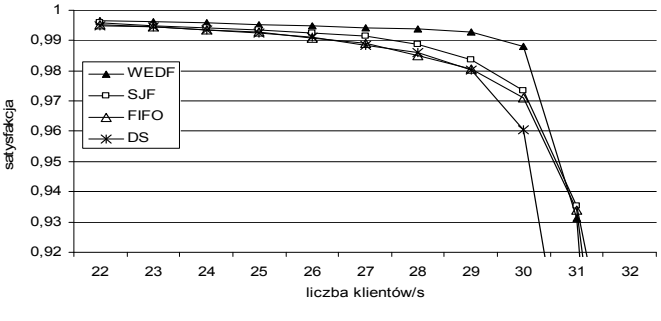
b)



c)

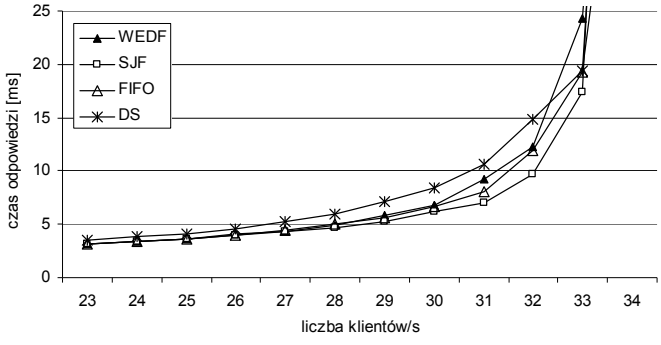


d)

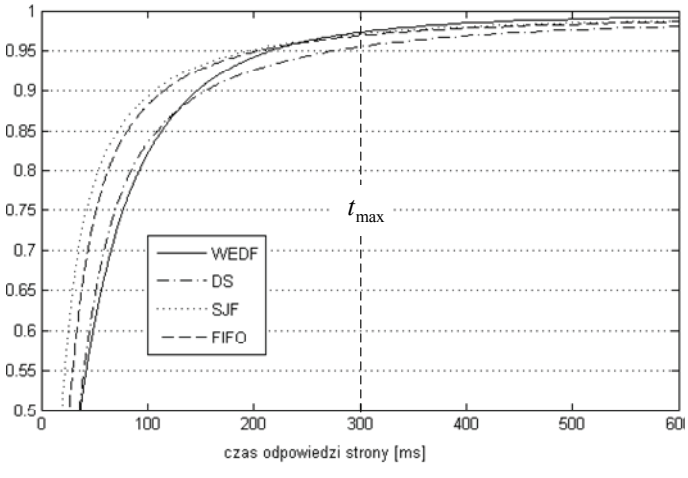


Rys. 4.8. Średnia wartość satysfakcji w funkcji obciążenia dla wariantów:
a) $t^s_{\max} = 300$ ms, $t^h_{\max} = 600$ ms, b) $t^s_{\max} = 400$ ms, $t^h_{\max} = 800$ ms,
c) $t^s_{\max} = 500$ ms, $t^h_{\max} = 1000$ ms, d) $t^s_{\max} = 500$ ms, $t^h_{\max} = 1000$ ms
dla wydłużonych czasów obsługi żądań na serwerze bazodanowym

a)



b)



Rys. 4.9. a) Średnie czasy odpowiedzi na żądanie w funkcji liczby nowych klientów dla wariantu $t^s_{\max} = 300$ ms, $t^h_{\max} = 600$ ms, b) dystrybuanta czasów odpowiedzi strony przy obciążeniu 32 nowych klientów/s dla wariantu $t^s_{\max} = 300$ ms, $t^h_{\max} = 600$ ms

Przejdźmy jeszcze do ostatniego z prezentowanych wykresów, do dystrybuanty czasów odpowiedzi (rys. 4.9b). Z wykresu tego wynika, że czasy odpowiedzi dla większości pobieranych stron są najdłuższe dla metody WEDF, czasy te są jednak do przyjęcia dla użytkowników. Jakość metody szeregowania uwidacznia się natomiast dla pewnej grupy stron, których czasy obsługi są długie (na rysunku wykres dystrybuanty metody WEDF przechodzi nad pozostałymi wykresami), w konsekwencji dla metody WEDF jest mniej stron, których czasy obsługi są naprawdę długie i przekraczają czas $t^s_{\max}$.

Zostały przeprowadzone dodatkowe badania i analizy mające na celu określenie czasu podjęcia decyzji, czasu jej realizacji w serwerze szeregującym oraz pesymistycznej czasowej złożoności obliczeniowej algorytmu. W badaniach nie były brane pod uwagę czasy nawiązania połączenia TPC/IP z klientem

i czasy przekazania żądania do serwera WWW. W tabeli 4.1 zestawione zostały średnie czasy obsługi żądania przez serwer szeregujący i maksymalna liczba obsłużonych żądań w ciągu sekundy. Badania prowadzone były na komputerze opisanym w rozdz. 3.1.3 wykorzystanym do badania wydajności przełącznika webowego.

Tabela 4.1

Liczba decyzji podjętych w ciągu sekundy i średni czas podjęcia decyzji dla różnych algorytmów szeregowania żądań

Algorytm	**Średnia liczba decyzji podjętych w ciągu sekundy**	**Średni czas podjęcia decyzji**
WEDF	40 ml	0,000025 ms
SJF	50 ml.	0,00002 ms
FIFO	57 ml.	0,000017 ms
DS	—	—

Jak wynika z badań, wydajność serwera szeregującego jest dość duża i jest większa niż możliwości obsługi nawet dużych serwisów webowych zawierających jeden serwer WWW, również czasy podjęcia decyzji i obsługi żądań w serwerze szeregującym są o kilka rzędów niższe od czasów obsługi żądań w serwerze WWW.

Przejdźmy teraz do oszacowania pesymistycznej czasowej złożoności obliczeniowej algorytmu podejmowania decyzji. Na czas podjęcia decyzji składają się następujące istotne czasy: czasy identyfikacji i wyznaczenia klasy żądanego obiektu, czas wyznaczenia obiektów nie pobranych w ramach strony WWW, czas wyznaczenia terminu d_i oraz czas wyznaczenia nowego uszeregowania na podstawie starego. Złożoność obliczeniowa związana z czasem identyfikacji i wyznaczenia klasy żądanego obiektu wynosi $O(\log N)$, gdzie N jest liczbą rodzajów obiektów pobieranych w ramach serwisu. Złożoność obliczeniowa dla procesu wyznaczenia obiektów nie pobranych w ramach strony WWW wynosi $O(G)$, gdzie G jest liczbą obiektów zagnieżdżonych w stronie. Czas wyznaczenia terminu d_i jest stały, dlatego też $O(1)$. Czas wyznaczenia nowego uszeregowania na podstawie starego jest to problem wstawiania do uporządkowanej listy nowego elementu i zależy od liczby elementów w kolejce, w tym wypadku liczba elementów w przybliżeniu odpowiada liczbie żądanych równocześnie w serwisie obiektów ara, dlatego $O(ara)$. Reasumując, czasowa złożoność obliczeniowa związana z podjęciem decyzji i wyznaczeniem nowego uszeregowania w systemie WEDF wynosi $O(\log N + G + ara)$.

Ze względu na wyniki przeprowadzonych badań wydajności serwera dystrybucji żądań oraz analizy pesymistycznej czasowej złożoności obliczeniowej, w której występują zależności liniowe oraz logarytmiczne, można stwierdzić, że odpowiednio skonstruowany serwer szeregujący może wydajnie obsługiwać serwisy webowe i nie powinien stać się „wąskim gardłem" systemu.

Podsumowując otrzymane wyniki badań oraz wyniki prezentowane w [209, 212] stwierdzić można, że serwis webowy, w którym stosowana byłaby metoda WEDF może mieć pożądane przez użytkowników własności. Zastosowanie metody może przynieść korzyści właścicielom serwisu dzięki zachowaniu przez serwis pożądanych czasów odpowiedzi dla stron, nawet przy dużym obciążeniu serwisu graniczącym z obciążeniem maksymalnym, który serwis może przyjąć.

4.2. SYSTEM MLF SZEREGOWANIA I DYSTRYBUCJI ŻĄDAŃ HTTP W ŚRODOWISKU LOKALNIE ROZMIESZCZONYCH SERWERÓW WEBOWYCH

Kolejnym prezentowanym systemem jest MLF. System MLF zapewnia czasy odpowiedzi dla całych stron WWW nie większe od zadanej wartości i wykorzystuje w swej konstrukcji klaster serwerów webowych umieszczonych w jednej lokalizacji. Budowa systemu MLF podobna jest do budowy opisanego wcześniej systemu LFNRD. Systemy te różnią się jednak sposobem działania przełącznika webowego. W systemie MLF przełącznik nie tylko dystrybuuje żądania, lecz również szereguje je w odpowiedni sposób.

Na wstępie opisane zostaną założenia dotyczące pracy systemu MLF, następnie przedstawiony zostanie projekt przełącznika MLF, a na końcu zaprezentowane zostaną wyniki badań.

4.2.1. OPIS SYSTEMU MLF

W skład systemu MLF wchodzą: klienci stanowiący źródło żądań HTTP, przełącznik webowy nazywany również dalej przełącznikiem MLF oraz klaster serwerów WWW wraz z serwerami zaplecza, w tym aplikacji i bazodanowymi. Sposób działania i funkcjonalności poszczególnych elementów w systemie determinuje metoda MLF opracowana dla systemu MLF.

Przyjmuje się, że wszystkie serwery webowe pracujące w klastrze mogą obsłużyć każde żądanie HTTP x_i takie, że $x_i \in X$, $i = 1,2,\ldots$.

Podobnie jak przy metodzie WEDF celem stosowania metody MLF jest taka obsługa żądań HTTP, aby czasy odpowiedzi $\dot{t}_{p_i}$ dla stron nie przekraczały maksymalnego czasu odpowiedzi $t_{\max}$.

Definicja 4.3. Czas odpowiedzi strony w systemie MLF jest czasem liczonym od momentu odebrania przez przełącznik webowy pierwszego żądania HTTP dotyczącego danej strony wysłanego przez danego klienta do momentu otrzymania przez przełącznik odpowiedzi HTTP, dotyczącej żądania o ostatni obiekt strony wysłanego przez tego samego klienta, przy czym czas odpowiedzi strony

pomniejszony jest o czasy, gdy w systemie nie znajdowało się w obsłudze żadne żądanie danego klienta dotyczące danej strony, a występujące między momentem otrzymania przez serwer szeregujący pierwszego i ostatniego żądania dotyczącego danej strony.

Czas odpowiedzi na żądanie HTTP w systemie MLF jest definiowany w taki sam sposób jak czas odpowiedzi na żądanie HTTP w systemie LFNRD (definicja 3.1).

Proces obsługi żądań w systemie MLF jest podobny od procesu obsługi żądań w systemie LFNRD. Klient po otrzymaniu adresu IP serwisu webowego przesyła do przełącznika MLF żądanie HTTP. Przełącznik wstawia otrzymane żądanie do kolejki, a następnie, gdy zostaną spełnione odpowiednie warunki, podejmuje decyzje o przekierowaniu żądania i przesyła je do wybranego serwera WWW. Serwer obsługuje żądanie i odsyła odpowiedź do przełącznika, a ten dalej do klienta.

Kluczowym elementem systemu MLF jest przełącznik MLF, który steruje systemem webowym. Przełącznik ten jest połączeniem przełącznika LFNRD oraz serwera szeregującego WEDF. W celu opisania działania przełącznika MLF przyjmuje się następujące dodatkowe oznaczenia:

d_i' – termin, gdy obsługa żądania powinna się zakończyć i odpowiedź na żądanie powinno w całości zostać przesłane do przełącznika,

$\Delta d_i'$ – maksymalny czas, który może być przeznaczony na obsługę żądania x_i przez serwer webowy.

Zadanie projektowe: należy opracować projekt przełącznika webowego MLF, w którym decyzje podejmowane są w dwóch etapach. W pierwszym etapie dla każdego nadchodzącego żądania x_i, gdzie $i = 1,2,...$, przy założeniu, że $ar(\tau_i^{(1)}) = ar_{\max}$, należy wyznaczyć uszeregowanie dla kolejki żądań HTTP Q_i działającej zgodnie z polityką EDF, w której wybór kolejności żądań realizowany jest na podstawie przyporządkowanych do żądań x_i terminów d_i, gdzie termin d_i jest tak wyznaczany, aby $\dot{t}_{p_i} \le t_{\max}$ przy założeniach, że $\neg \underset{p_i}{\exists}\left[\left(\dot{t}_{p_i} > t_{\max}\right) \wedge \left(\underset{i=1,2...}{\forall} ara(\tau_i^{(1)}) = ar_{\max}\right)\right]$ oraz $\underset{x_i}{\forall}\left(ara(\tau_i^{(1)}) < ara_{\max}\right)$. W przypadku, gdy $ar(\tau_i^{(1)}) < ar_{\max}$, żądanie x_i nie jest umieszczane w kolejce Q_i.

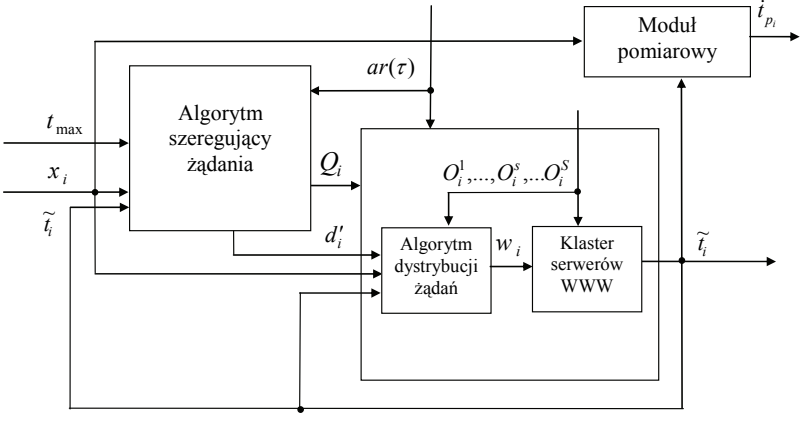
W drugim etapie dla każdego opuszczającego kolejkę żądania x_i lub żądania, które w kolejce nie było umieszczone, w oparciu o wiedzę o obciążeniach serwerów webowych $O_i^1,...,O_i^s,...,O_i^S$ oraz wiedzę o przeszłych czasach odpowiedzi na żądania $\tilde{t}_1,...,\tilde{t}_{i-1}$, należy wyznaczyć serwer webowy w_i spośród S serwerów, dla którego spełniony jest warunek $\hat{t}_i^{w_i} \le d_i' - \tau_i^{(2)}$, o ile

$\exists_{s\in\{1,...,S\}} \hat{t}_i^s \le d_i' - \tau_i^{(2)}$ i $ar(\tau_i^{(1)}) = ar_{\max}$ lub w przeciwnym wypadku $w_i : \hat{t}_i^{w_i} = \min\left\{\hat{t}_i^s : s = 1,...,S\right\}$.

Na rys. 4.10 przedstawiony został ogólny schemat procesu podejmowania decyzji w systemie MLF, w którym wyróżnić można odpowiedni algorytm szeregujący wyznaczający kolejkę Q_i i termin d_i' oraz algorytm dystrybucji żądań wyznaczający serwer WWW w klastrze do obsługi żądania HTTP.

Uogólniając, należy zaproponować konstrukcję przełącznika webowego, który będzie odpowiednio szeregował żądania oraz je dystrybuował pomiędzy serwery WWW. Szeregowanie żądań powinno odbywać się w taki sposób, aby czasy odpowiedzi dla stron były krótsze od żądanego czasu $t_{\max}$ przy założeniu, że przy dużym obciążeniu czas $t_{\max}$ jest osiągalny dla wszystkich stron. Należy więc wyznaczać odpowiednio terminy d_i, $i = 1,2,...$, tak, aby osiągnąć zamierzony cel.

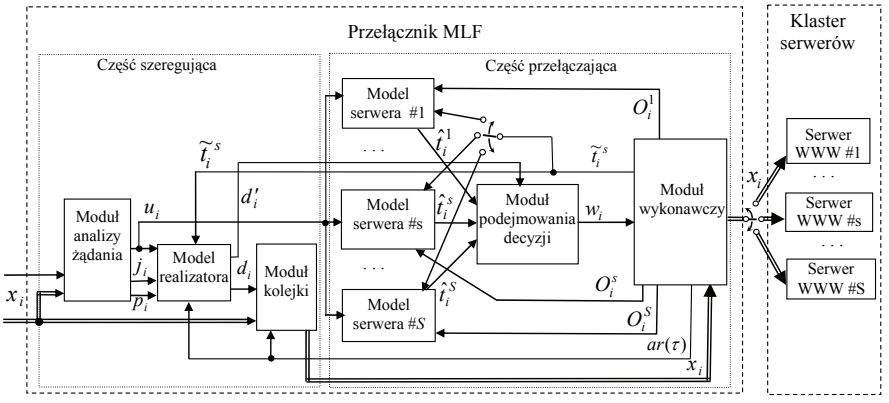
W ramach dystrybucji żądań HTTP pomiędzy serwery webowe przełącznik powinien wybierać do obsługi te serwery, które oferują takie czasy obsługi żądań, aby obsługa zakończyła się przed terminami d_i', $i = 1,2,...$



Rys. 4.10. Schemat prezentujący proces podejmowania decyzji w systemie MLF

4.2.2. PRZEŁĄCZNIK MLF

Przełącznik MLF składa się z dwóch części: szeregującej oraz przełączającej. W części szeregującej wyróżnić można: moduł analizy żądania, model realizatora, moduł kolejki. W części przełączającej znajdują się: modele serwerów, moduł podejmowania decyzji i moduł wykonawczy. Na rys. 4.11 przedstawiony jest schemat przełącznika MLF.



Rys. 4.11. Schemat przełącznika MLF

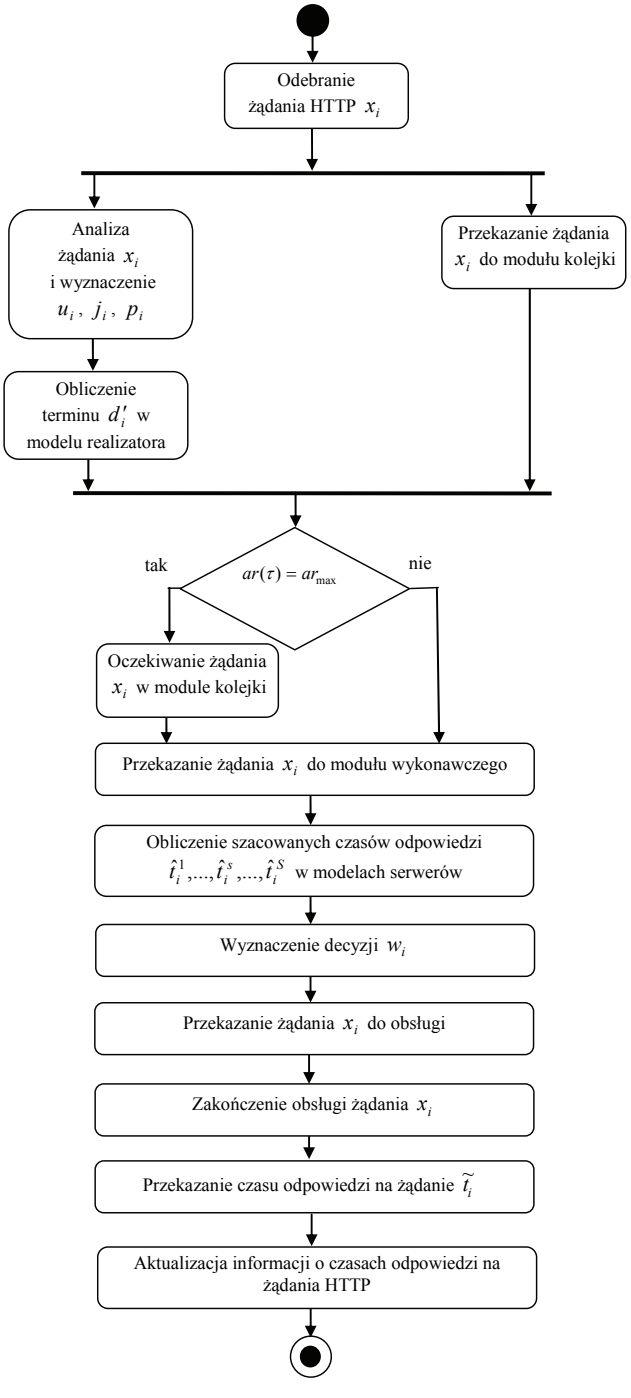
Część szeregująca szereguje żądania tworząc kolejkę Q_i w taki sposób, aby dla każdej strony nie dopuszczać do przekroczenia czasu $t_{\max}$.

W **części przełączającej** wyznaczana jest decyzja w_i o przekierowaniu żądania w taki sposób, by czas zakończenia obsługi żądania x_i był zbliżony do wyznaczonego w części szeregującej terminu d_i'.

Na rys. 4.12 przedstawiony został diagram czynności ilustrujący proces obsługi żądania w przełączniku MLF.

Przełącznik po otrzymaniu żądania HTTP x_i analizuje je w module analizy żądania, wyodrębnia adres żądanego obiektu u_i, identyfikator użytkownika j_i oraz identyfikator strony p_i, w ramach której dany obiekt jest żądany (dwie ostatnie informacje powinny być zawarte w polu *cookie* żądania). Następnie model realizatora wyznacza termin d_i, w którym powinna rozpocząć się obsługa żądania oraz termin d_i' wskazujący na moment, w którym obsługa żądania powinna się zakończyć. W dalszej kolejności żądanie jest wstawiane do modułu kolejki Q_i, o ile $ar(\tau_i^{(1)}) = ar_{\max}$. Kolejka Q_i w przełączniku MLF definiowana jest w taki sam sposób jak kolejka w serwerze szeregującym WEDF opisana w rozdz. 4.1.2 tak, że żądania w kolejce uszeregowane są zgodnie z wyznaczonymi terminami d_i. Jeżeli $ar(\tau_i^{(1)}) < ar_{\max}$, to żądanie zostaje przesłane dalej bezpośrednio do modułu wykonawczego, bez umieszczenia w kolejce.

Wartość $ar_{\max}$ podobnie jak w metodzie WEDF, powinna być równa minimalnej liczbie żądań, dla której średnia przepustowość systemu osiąga wartość maksymalną.



Rys. 4.12. Diagram czynności obsługi żądania w przełączniku MLF

Żądanie x_i opuszcza kolejkę Q_i, gdy wszystkie żądania umieszczone w kolejce przed nim opuszczą kolejkę oraz gdy wystąpi zdarzenie w pewnym momencie $\tau_i^{(2)}$ takie, że liczba obsługiwanych aktywnie żądań zmniejszy się, spełniając warunek $ar(\tau_i^{(2)}) < ar_{\max}$.

Po opuszczeniu kolejki żądanie x_i jest przekazywane do części przełączającej przełącznika MLF, a dokładniej do modułu wykonawczego. Na podstawie informacji o obciążeniach serwerów w klastrze $O_i^1,...,O_i^s,...,O_i^S$ oraz informacji o adresie żądanego obiektu, moduły modeli serwerów wyliczają szacowane czasy odpowiedzi $\hat{t}_i^1,...,\hat{t}_i^s,...,\hat{t}_i^S$ na żądanie dla poszczególnych serwerów WWW. Oszacowane czasy odpowiedzi przesyłane są do modułu podejmowania decyzji. Moduł ten wyznacza decyzję w_i, czyli wskazuje numer serwera, który obsłuży żądanie. Decyzja jest przekazywana do modułu wykonawczego, który przesyła żądanie x_i do wybranego serwera WWW. Po zakończeniu obsługi żądania moduł wykonawczy przekazuje zmierzony czas odpowiedzi $\tilde{t}_i$ do modułu modelu serwera, który odpowiada serwerowi wykonującemu obsługę żądania oraz do modułu modelu realizatora, o ile żądanie wcześniej zostało umieszczone w kolejce w module kolejki. Oba moduły na podstawie otrzymanych czasów aktualizują informacje niezbędne do szacowania czasów obsługi i wyznaczenia terminu obsługi.Opisane zostanie teraz działanie poszczególnych modułów przełącznika. **Moduł analizy żądania oraz moduł kolejki** w przełączniku MLF działają w taki sam sposób jak moduł analizy żądania i moduł kolejki w serwerze szeregującym opisanym w rozdz. 4.1.2.

Model realizatora działa podobnie jak model realizatora opisany w rozdz. 4.1.2 z drobną jednak różnicą. W modelu realizatora wyznaczany jest dodatkowo termin d'_i wskazujący na moment, gdy obsługa i-tego żądania powinna się zakończyć i odpowiedź na żądanie powinna znaleźć się w module wykonawczym przełącznika. Termin ten wyliczany jest zgodnie z zależnością:

$$d_i' = \tau_i^{(1)} + \Delta d_i, \tag{4.5}$$

gdzie sposób obliczania Δd_i opisany jest zależnością (4.1).

Modele serwerów w przełączniku MLF mają taką samą konstrukcję jak w przełączniku LFNRD opisanym w rozdz. 3.1.2. Modele serwerów są modelami systemu obsługi żądań HTTP zawierającymi w swej konstrukcji sieć rozmyto-neuronową. Każdy z modeli szacuje czas $\hat{t}_i^s$ na podstawie adresu żądanego obiektu u_i oraz obciążenia $O_i^s = \left[a_i^s, b_i^s\right]$, gdzie $a_i^s = ar_i^s$ jest obciążeniem s-tego serwera WWW i jest równe liczbie równocześnie obsługiwanych żądań przez ten serwer, $b_i^s = bd_i^s$ jest liczbą równocześnie obsługiwanych żądań

HTTP dotyczących obiektów dynamicznych. Model serwera aktualizuje swoje informacje o serwerze na podstawie zmierzonego czasu $\tilde{t}_i$ obsługi żądania x_i.

Moduł podejmowania decyzji działa w inny sposób niż w przełączniku LFNRD i stanowi kluczowy element w konstrukcji przełącznika MLF. Moduł ten wyznacza decyzję w_i wskazując serwer, do którego zostanie wysłane żądanie x_i. W celu wyznaczenia decyzji obliczany jest maksymalny pozostały czas $\Delta d_i'$, jaki może być przeznaczony na obsługę żądania x_i:

$$\Delta d_i' = d_i' - \tau_i^{(2)}. \tag{4.6}$$

Decyzja w_i wyznaczana jest zgodnie z zależnością:

$$w_i = \begin{cases} \min\left\{s : s \in \{1,...,S\} \wedge \Delta d_i' \le \hat{t}_i^s\right\} \text{ gdy } ar\left(\tau_i^{(1)}\right) = ar_{\max} \text{ i } \exists\limits_{s\in\{1,...,S\}} \Delta d_i' \le \hat{t}_i^s \\ s_{\min} : \hat{t}_i^{s_{\min}} = \min\left\{\hat{t}_i^s : s = 1,...,S\right\} \text{ dla pozostalych przypadków} \end{cases} \tag{4.7}$$

W procesie decyzyjnym wyznaczany jest serwer webowy o najmniejszym numerze, który oferuje czas obsługi $\hat{t}_i^s$ nie dłuższy niż pozostały czas $\Delta d_i'$ przeznaczony na obsługę żądania. Jeśli nie istnieje serwer, dla którego jest spełniony ten warunek lub też żądanie x_i nie zostało wstawione wcześniej do kolejki Q_i, ponieważ $ar\left(\tau_i^{(1)}\right) < ar_{\max}$, to decyzja podejmowana jest podobnie jak w przełączniku LFNRD i wybierany jest serwer, dla którego szacowany czas odpowiedzi jest najkrótszy.

Decyzje są tak podejmowane, aby żądania były najczęściej obsługiwane przez serwery o najniższych numerach, serwery te powinny być w związku z tym najbardziej obciążone. Z kolei serwery o najwyższych numerach powinny pozostać nieobciążone, a tym samym powinny umożliwiać szybką obsługę dla żądań, które takiej obsługi wymagają.

Moduł wykonawczy, podobnie jak w przełączniku LFNRD przesyła żądanie HTTP do obsługi do wybranego serwera WWW. W trakcie obsługi moduł ten mierzy rzeczywisty czas $\tilde{t}_i$ obsługi żądania. Również przekazuje informacje o liczbie $ar(\tau)$ obsługiwanych równocześnie żądań oraz o obciążeniach $O_i^1,...,O_i^s,...,O_i^S$ poszczególnych serwerów webowych.

Jak wynika z zamieszczonego powyżej opisu, przełącznik MLF stanowi hybrydę serwera szeregującego oraz przełącznika LFNRD. Na wejściu do przełącznika żądania HTTP są odpowiednio szeregowane i wstawiane do kolejki, a przy wyznaczaniu terminów d_i, reszta systemu traktowana jest jak czarna skrzynka, której struktura i sposób działania nie są znane, znane są natomiast jedynie czasy odpowiedzi na żądania. Ta część przełącznika jest częścią szeregującą. Po opuszczeniu modułu kolejki żądanie trafia do części przełączającej, która wybiera serwer WWW do obsługi żądania. Przełącznik ten podobnie jak prze-

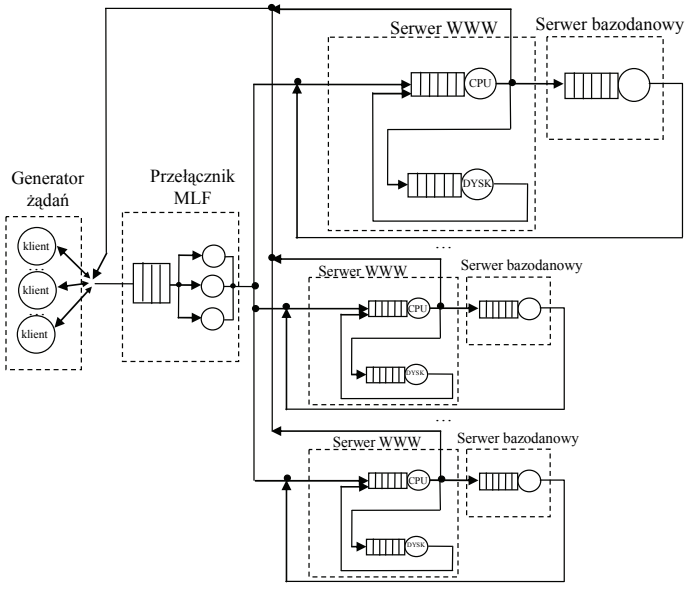
łącznik LFNRD szacuje czasy odpowiedzi na żądanie wybierając jednak nie serwer, który obsłuży żądanie najszybciej, ale ten, który jest najbardziej obciążany i oferuje czas obsługi krótszy niż wymagany. W ten sposób pozostawia się część serwerów nieobciążonych, które będą w stanie obsłużyć żądania, gdy żądany czas odpowiedzi będzie krótki.

Metoda MLF może być wykorzystywana zarówno dla klastrów serwerów homogenicznych jak i heterogenicznych. W przypadku serwerów heterogenicznych klaster powinien być tak skonfigurowany, aby serwery o najmniejszej mocy obliczeniowej miały przydzielone numery o najmniejszych wartościach, natomiast o najwyższej mocy obliczeniowej miały przydzielone numery o najwyższych wartościach.

4.2.3. BADANIA DOTYCZĄCE SYSTEMU MLF

W celu oceny własności systemu MLF przeprowadzone zostały badania symulacyjne. Program symulacyjny składał się z następujących modułów: modułu generatora, modułu przełącznika MLF, modułu serwera WWW i modułu serwera bazodanowego.

Na rys. 4.13 przedstawiony został schemat zastosowanego modelu symulacyjnego. Schemat ten podobny jest do schematu zastosowanego przy badaniu metody LFNRD.



Rys. 4.13. Schemat modelu symulacyjnego wykorzystanego w badaniach symulacyjnych nad systemem MLF

Moduł generatora żądań w programie symulacyjnym został skonstruowany w taki sam sposób jak moduł generatora opisany w rozdz. 4.1.4. Natomiast moduł serwera WWW i serwera bazodanowego został skonstruowany w taki sposób jak zostało to opisane w rozdz. 3.1.3.

Moduł przełącznika MLF w programie symulacyjnym pracował w sposób opisany w rozdz. 4.2.2. Dodatkowo w module przełącznika zaimplementowane zostały inne algorytmy dystrybucji żądań, które są obecnie stosowane w rozwiązaniach przemysłowych lub znane są z literatury. Do zaimplementowanych metod i algorytmów należały: MLF, LFNRD, LARD, CAP, RR.

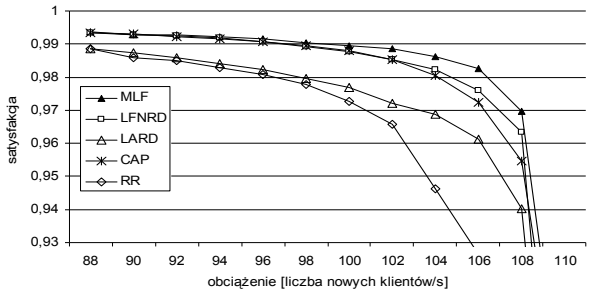
W trakcie prowadzonych eksperymentów symulacyjnych wyliczane były średnie wartości satysfakcji, średnie wartości czasów odpowiedzi na żądania HTTP, czasy odpowiedzi dla całych stron. Eksperymenty symulacyjne przeprowadzone zostały dla czterech różnych wariantów konfiguracji serwerów i żądanych czasów odpowiedzi dla stron. W pierwszych dwóch wariantach wykorzystany został klaster zawierający trzy homogeniczne serwery WWW i bazodanowe (klaster ten oznaczony został na rysunkach Hom3s). W pierwszym wariancie prowadzone były badania dla czasów $t^s_{\max} = 300ms$ i $t^h_{\max} = 600ms$ (wartość $t^h_{\max}$ podobnie jak w eksperymentach prowadzonych dla metody WEDF była dwukrotnie większa niż $t^s_{\max}$).

W drugim wariancie przyjęte zostały czasy $t^s_{\max} = 500ms$ i $t^h_{\max} = 1000ms$. W trzecim wariancie klaster zawierał pięć homogenicznych serwerów WWW i bazodanowych przy przyjętych czasach $t^s_{\max} = 300ms$ i $t^h_{\max} = 600ms$ (klaster oznaczony został na rysunkach Hom5s). W ostatnim wariancie wykorzystany został klaster heterogeniczny zawierający trzy serwery WWW i bazodanowe, przy czym jeden zestaw serwera WWW i bazodanowego miał wydłużone wszystkie czasy obsługi żądań o 33% na poszczególnych stanowiskach obsługi (oznaczenie na rysunkach Het1s/2s). Dla tego wariantu przyjęte zostały czasy $t^s_{\max} = 300ms$ i $t^h_{\max} = 600ms$.

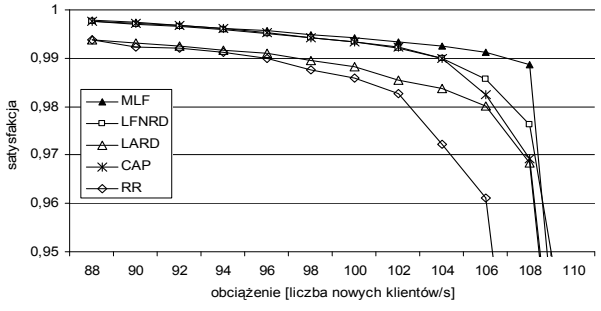
Na rys. 4.14 przedstawione zostały wykresy średnich wartości satysfakcji w funkcji obciążenia serwisu, czyli liczby nowych klientów przybywających na sekundę. Na rys 4.15a przedstawione zostały wykresy średnich czasów odpowiedzi na żądania HTTP w funkcji obciążenia natomiast na rys. 4.15b zaprezentowane zostały wykresy dystrybuanty czasów odpowiedzi dla stron dla poszczególnych algorytmów dystrybucji żądań.

Z wykresów na rys. 4.14 wynika, że największe wartości satysfakcji, zwłaszcza przy większym obciążeniu, uzyskane zostały dla metody MLF. Przy małym obciążeniu dla wszystkich metod i algorytmów średnie wartości satysfakcji były zazwyczaj powyżej 0,98.

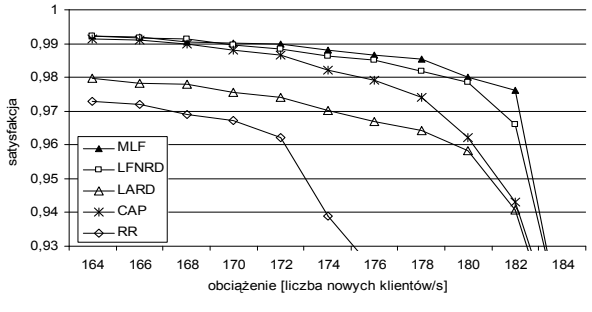
a)



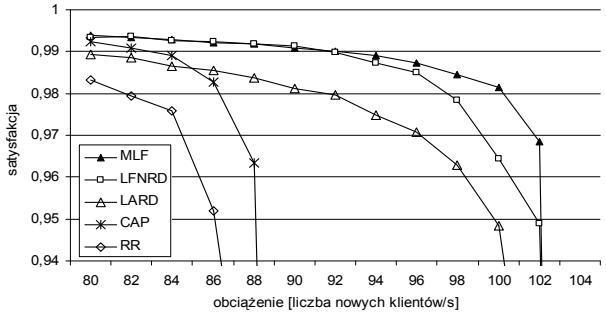
b)



c)

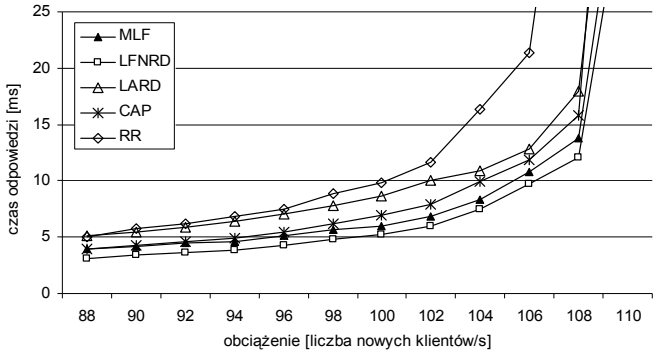


d)

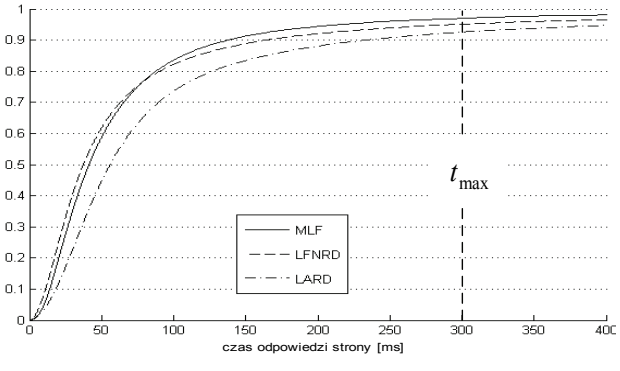


Rys. 4.14. Średnia wartość satysfakcji w funkcji obciążenia dla wariantów:
a) $t^s_{\max} = 300ms$, $t^h_{\max} = 600ms$, Hom3s, b) $t^s_{\max} = 500ms$, $t^h_{\max} = 1000ms$, Hom3s,
c) $t^s_{\max} = 300ms$, $t^h_{\max} = 600ms$, Hom5s, d) $t^s_{\max} = 300ms$, $t^h_{\max} = 600ms$, Het1s/2s

a)



b)



Rys. 4.15. a) Średnie czasy odpowiedzi na żądania w funkcji liczby nowych klientów dla wariantu $t^s_{\max} = 300ms$, $t^h_{\max} = 600ms$, Hom3s, b) Dystrybuanta czasów odpowiedzi dla strony przy obciążeniu 100 klientów/s dla wariantu $t^s_{\max} = 300ms$, $t^h_{\max} = 600ms$, Het1s/2s

Dobre wyniki dla metody MLF zostały otrzymane zarówno dla homogenicznych klastrów serwerów webowych jak i klastra heterogenicznego.

Z wykresów prezentowanych na rys. 4.15a wynika, że najkrótsze czasy odpowiedzi na żądania HTTP w badaniach dla klastra Hom3s uzyskane zostały dla algorytmu LFNRD. Wnioskować więc można, że uzyskanie krótkich czasów odpowiedzi na pojedyncze żądania nie zapewnia uzyskania wysokiej satysfakcji. Podobne wyniki uzyskane zostały dla metody szeregowania WEDF.

Ostatni z prezentowanych wykresów (rys. 4.15b) przedstawia dystrybuanty czasów odpowiedzi dla całych stron. Wyniki przedstawione zostały dla metody MLF oraz algorytmów dystrybucji LFNRD i LARD (dla pozostałych algorytmów system przy danym obciążeniu był przeciążony). Jak wynika z wykresu dystrybuanta dla metody MLF przechodzi nad wykresami pozostałych dystrybuant dla czasu odpowiedzi stron dłuższych niż 75 ms. Oznacza to, że dla metody MLF najmniejsza liczba stron uzyskała czasy odpowiedzi dłuższe niż 75 ms.

Podobnie jak dla pozostałych algorytmów opisanych w poprzednich rozdziałach przeprowadzone zostały badania, mające na celu określenie wydajności

przełącznika MLF oraz obliczona została pesymistyczna czasowa złożoność obliczeniowa algorytmów podejmowania decyzji. Ponieważ wyniki badań wydajności dla pozostałych algorytmów dystrybucji żądań zostały zaprezentowane w tabeli 3.3, dlatego przedstawione zostaną jedynie wyniki dla metody MLF. Liczba żądań obsługiwanych na sekundę przez komputer opisany w rozdz. 3.1.3 dla metody MLF wynosi 606 tysięcy żądań na sekundę, natomiast średni czas odpowiedzi na żądanie wynosi 0,00165 ms. Liczba obsługiwanych żądań dla przełącznika MLF jest nieznacznie mniejsza niż dla przełącznika LFNRD.

Ponieważ w przełączniku MLF wykorzystywane są dwa algorytmy podejmowania decyzji, w których decyzje podejmowane są w różnych momentach, dlatego złożoność obliczeniowa powinna być dla nich wyznaczona osobno. Złożoność obliczeniową algorytmu stosowanego w części szeregującej można wyznaczyć w podobny sposób jak dla systemu WEDF i wynosi $O(\log N + G + ara)$, gdzie N jest liczbą rodzajów obiektów pobieranych w ramach serwisu, G jest liczbą obiektów zagnieżdżonych w stronie, ara jest liczbą żądań równocześnie obsługiwanych. Złożoność obliczeniową algorytmu stosowanego w części przełączającej można wyznaczyć w podobny sposób jak dla systemu LFNRD i wynosi ona $O(S)$, gdzie S jest liczbą serwerów WWW pracujących w klastrze.

Ze względu na wyniki przeprowadzonych badań wydajności przełącznika MLF oraz analizy pesymistycznej czasowej złożoności obliczeniowej, w której występują zależności liniowe oraz logarytmiczne, można stwierdzić, że odpowiednio skonstruowany przełącznik może wydajnie obsługiwać serwis webowy i nie powinien stać się „wąskim gardłem" systemu.

Po przeprowadzeniu badań symulacyjnych opisanych powyżej oraz prezentowanych w [210] wnioskować można, że dzięki zastosowaniu metody MLF w lokalnych klastrach webowych możliwe jest zapewnienie czasu odpowiedzi nie dłuższego niż zadany dla całych stron WWW, nawet przy dużym obciążeniu serwisu, graniczącym z obciążeniem maksymalnym, który serwis może przyjąć.

4.3. SYSTEM GGARDiB DYSTRYBUCJI I SZEREGOWANIA ŻĄDAŃ HTTP W ŚRODOWISKU GLOBALNIE ROZMIESZCZONYCH SERWERÓW WEBOWYCH Z SERWERAMI POŚREDNICZĄCYMI

Jako ostatni opisany zostanie system GGARDiB. System ten przeznaczony jest do pracy w rozległych sieciach komputerowych obsługujących dużą liczbę klientów webowych rozmieszczonych w różnych rejonach świata. System zapewnia obsługę całych stron WWW w taki sposób, aby czasy odpowiedzi dla stron nie przekraczały ustalonego czasu $t_{\max}$.

Budowa systemu GGARDiB podobna jest do budowy systemu GARDiB. W systemie GGARDiB również wykorzystywane są serwery pośredniczące oraz serwisy lokalne, jednak sposób ich działania jest odmienny niż w systemie GARDiB. O ile w systemie GARDiB serwis lokalny może być klastrem serwerów webowych, którego struktura i sposób działania nie są narzucane w ramach metody GARDiB, o tyle w systemie GGARDiB sposób działania przełącznika webowego jest ściśle określony ze względu na potrzebę jego współdziałania z serwerem pośredniczącym.

W rozdziale przedstawione zostaną główne założenia dotyczące działania systemu GGARDiB, zaprezentowane zostaną projekty serwera pośredniczącego oraz przełącznika webowego, a na zakończenie omówione zostaną wyniki przeprowadzonych badań.

4.3.1. OPIS SYSTEMU GGARDiB

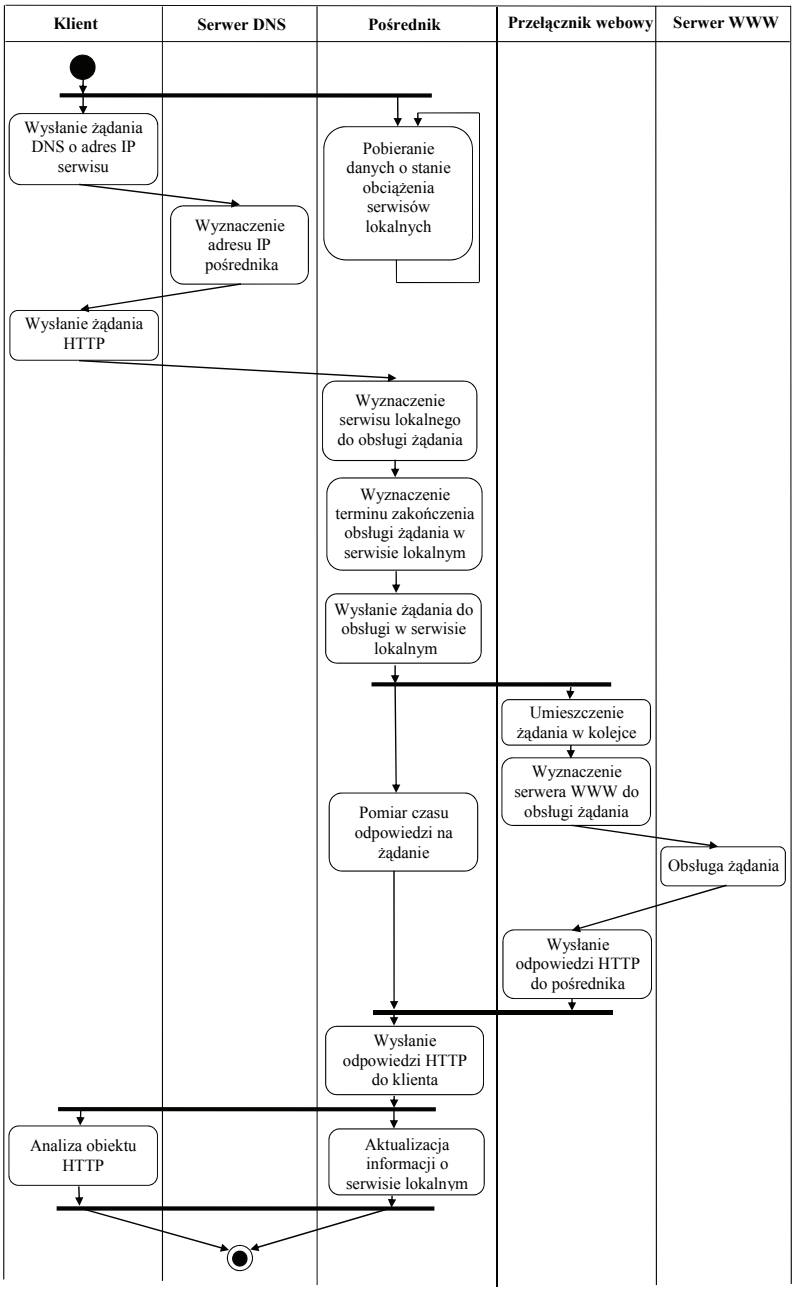
W systemie GGARDiB wyróżnić można następujące elementy: serwisy lokalne, serwery pośredniczące nazywane również dalej pośrednikami, serwer lub serwery DNS, klientów wysyłających żądania HTTP. Sposób połączenia poszczególnych elementów jest taki sam, jak w systemie GARDiB i przedstawiony został na rys. 3.8. Działanie i funkcjonalności elementów w systemie GGARDiB determinuje metoda GGARDiB.

Diagram czynności obrazujący sposób obsługi żądań HTTP w systemie GGARDiB przedstawiony jest na rys. 4.16. Diagram ten jest podobny dla diagramu czynności dla systemu GARDiB (rys. 3.9).

Na początku klient wysyła żądanie DNS do systemu DNS. Autorytatywny serwer DNS w odpowiedzi przekazuje adres IP serwera pośredniczącego, który znajduje się w najbliżej odległości geograficznej od klienta. Klient przesyła do wskazanego pośrednika żądanie HTTP. Pośrednik, posiadając wiedzę o obciążeniu poszczególnych serwisów lokalnych i obciążeniu łączy na drodze do serwisów lokalnych, wybiera serwis lokalny, dla którego oszacowany czas odpowiedzi jest najkrótszy. Dodatkowo pośrednik wyznacza termin zakończenia obsługi żądania w serwisie lokalnym. Termin ten jest tak wyznaczony, aby czas odpowiedzi dla całej strony nie był większy niż zadana wartość $t_{\max}$.

Następnie pośrednik wysyła do wybranego serwisu lokalnego żądanie HTTP otrzymane od klienta. Przełącznik webowy w serwisie lokalnym po otrzymaniu żądania wstawia żądania do kolejki, biorąc pod uwagę wyznaczony przez pośrednika termin zakończenia obsługi żądania w serwisie lokalnym. W dalszej kolejności przełącznik webowy wybiera zgodnie z algorytmem dystrybucji żądań serwer webowy, który obsłuży żądanie HTTP i przesyła żądanie do wybranego serwera. W momencie, gdy odpowiedź HTTP jest gotowa, przygotowany obiekt HTTP jest przesyłany do pośrednika. Pośrednik mierzy czas odpowiedzi na żądanie HTTP i po otrzymaniu odpowiedzi HTTP w całości

przesyła ją do klienta, a następnie modyfikuje w procesie adaptacji swoje informacje o serwisie lokalnym, który obsłużył żądanie.



Rys. 4.16. Diagram czynności prezentujący obsługę żądania w systemie GGARDiB

Definicja 4.5. Czasem odpowiedzi strony w systemie GGARDiB jest czas liczony od momentu pojawienia się w serwerze pośredniczącym pierwszego żądania HTTP dotyczącego strony, a wysłanego przez klienta, do momentu otrzymania przez serwer pośredniczący odpowiedzi HTTP, dotyczącej żądania o ostatni obiekt strony wysłanego przez tego samego klienta, przy czym czas odpowiedzi strony pomniejszony jest o czasy, gdy w systemie nie znajdowało się w obsłudze żadne żądanie danego klienta dotyczące danej strony, a występujące między momentem otrzymania przez serwer szeregujący pierwszego i ostatniego żądania dotyczącego danej strony.

Czas odpowiedzi na żądanie w systemie GGARDiB definiowany jest tak samo jak w systemie GARDiB (definicja 3.2)

System GGARDiB różni się od systemu GARDiB budową i działaniem serwera pośredniczącego i przełącznika webowego, nazywanych również dalej pośrednikiem GGARDiB i przełącznikiem GGARDiB. Dlatego też ich budowa i działanie zostanie szczegółowo opisane.

Proces decyzyjny w systemie GGARDiB jest wieloetapowy i jest realizowany w serwerze pośredniczącym GGARDiB oraz przełączniku GGARDiB. W celu opisania budowy i działania pośrednika i przełącznika GGARDiB zostaną wprowadzone nowe oznaczenia oraz przedefiniowane oznaczenia już wprowadzone. Wskazane oznaczenia częściowo pokrywają się z oznaczeniami wprowadzonymi w poprzednich rozdziałach, jednak ze względu na opis pośrednika i przełącznika w jednym rozdziale zaistniała potrzeba precyzyjnego ich przedstawienia powtórnie.

x_i – żądanie HTTP,

x_i' – zmodyfikowane żądanie HTTP,

s'' – indeks serwisu lokalnego, $s'' \in \{1,...,S''\}$,

S'' – liczba serwisów lokalnych,

s – indeks serwera webowego, $s \in \{1,...,S^{s''}\}$,

$S^{s''}$ – liczba serwerów webowych w s''-tym serwisie lokalnym,

$\widetilde{t}_i''$ – zmierzony czas odpowiedzi na i-te żądanie w serwerze pośredniczącym GGARDiB,

$\hat{t}_i''^{s''}$ – szacowany czas odpowiedzi na i-te żądanie dla s''-tego serwisu lokalnego w serwerze pośredniczącym GGARDiB,

$\widetilde{t}_i$ – zmierzony czas odpowiedzi na i-te żądanie w przełączniku GGARDiB,

$\hat{t}_i^s$ – szacowany czas odpowiedzi na i-te żądanie dla s-tego serwera webowego w przełączniku GGARDiB,

$O_i''^{s''}$ – obciążenie charakteryzujące s''-ty serwis lokalny oraz łącze na kierunku od serwisu do pośrednika, $O_i''^{s''} = \left[arl_i^{s''}, btt_i^{s''}\right]$,

$\tau_i^{(1)}$ – moment nadejścia i-tego żądania do serwera pośredniczącego,

$\tau_i^{(2)}$ – moment nadejścia i-tego żądania do przełącznika webowego,

$\tau_i^{(3)}$ – moment, gdy i-te żądanie opuszcza kolejkę Q_i w przełączniku webowym,

$arl^{s''}(\tau)$ – obciążenie s''-tego serwisu lokalnego, jego wartość równa jest liczbie żądań znajdujących się w serwisie lokalnym (w kolejkach w przełączniku oraz w serwerach WWW) w momencie τ,

$arl_i^{s''}$ – obciążenie s''-tego serwisu lokalnego w momencie $\tau_i^{(1)}$, $arl_i^{s''} = arl^{s''}(\tau_i^{(1)})$,

$btt_i^{s''}$ – obciążenie sieci rozległej na kierunku od pośrednika do s''-tego serwisu lokalnego, jego wartość równa jest czasowi transferu próbnego obiektu od serwisu lokalnego do pośrednika, jest to najaktualniejsza wartość w momencie nadejścia żądania x_i,

$arl_{\max}^{s''}$ – liczba żądań w s''-tym serwisie lokalnym, powyżej której czasy odpowiedzi na żądania HTTP wzrastają ponad akceptowalną wartość,

$ars^{s''}(\tau)$ – liczba żądań obsługiwanych równocześnie w realizatorach (serwerach WWW) s''-tego serwisu lokalnego w momencie τ,

$ars_{\max}^{s''}$ – maksymalna liczba żądań obsługiwanych równocześnie przez wszystkie realizatory (serwery WWW) w s''-tym serwisie lokalnym,

O_i^s – obciążenie s-tego serwera webowego, $O_i^s = \left[ar_i^s, bd_i^s\right]$,

ar_i^s – element obciążenia s-tego serwera webowego, jego wartość jest równa liczbie równocześnie obsługiwanych żądań HTTP przez serwer,

bd_i^s – element obciążenia s-tego serwera webowego, jego wartość równa jest liczbie równocześnie obsługiwanych żądań HTTP dotyczących obiektów dynamicznych,

w_i'' – numer serwisu lokalnego wybranego do obsługi i-tego żądania, $w_i'' \in \{1,...,S''\}$,

w_i – numer serwera webowego wybranego do obsługi i-tego żądania, $w_i \in \{1,...,S\}$,

W_i'' – wektor wskazujący serwisy lokalne wybrane do obsługi żądań dotyczących obiektów zagnieżdżonych w stronie, $W_i'' = \left[w_{i1}'',...,w_{il}'',...,w_{iL}''\right]$,

w_{il}'' – element wektora W_i'', numer serwisu lokalnego wybranego do obsługi żądania dotyczącego l-tego obiektu należącego do strony, $w_i'' \in \{1,...,S''\}$,

p_i – identyfikator strony, w ramach której i-ty obiekt jest żądany,

$t_{\max}$ – zadany nieprzekraczalny czas odpowiedzi dla strony,

$\dot{t}_{p_i}$ – czas odpowiedzi całej strony o identyfikatorze p_i,

j_i – identyfikator użytkownika, który wysłał i-te żądanie,

db_i – termin, gdy i-te żądanie powinno zostać w całości przekazane z serwera webowego do przełącznika webowego GGARDiB,

Δdb_i – maksymalny czas, który może upłynąć od momentu wysłania i-tego żądania z serwera pośredniczącego, do momentu otrzymania w całości żądanego obiektu,

ds_i – termin, gdy obsługa i-tego żądania powinna rozpocząć się na serwerze webowym,

Δds_i – maksymalny czas, jaki może być poświęcony na obsługę i-tego żądania przez serwer webowy,

$h_i^{s''}$ – czas transmisji odpowiedzi na i-te żądanie z s''-tego serwisu lokalnego do serwera pośredniczącego,

v_i – wielkość odpowiedzi na i-te żądanie,

$\rho_i^{s''}$ – efektywna przepustowość na kierunku od s''-tego serwisu lokalnego do serwera pośredniczącego,

Z_i – wektor klas obiektów należących do strony, a nie pobranych jeszcze w ramach danej strony przez klienta $Z_i = \left[k_i^1,...,k_i^l,...,k_i^L\right]$,

k_i^l – element wektora Z_i, klasa jednego z obiektów nie pobranych jeszcze przez użytkownika w ramach danej strony, $l = 1,...,L$,

L – liczba elementów strony nie pobranych jeszcze przez użytkownika,

tp_{p_i} – czas obsługi strony do momentu nadejścia i-tego żądania,

U_i' – parametry realizatora $U_i' = \left[\hat{t}_{1i}',...,\hat{t}_{ki}',...,\hat{t}_{Ki}'\right]$,

$\hat{t}_{ki}'$ – składowa wektora U_i', szacowany czas odpowiedzi na żądanie należące do k-tej klasy,

K – liczba klas obiektów obsługiwanych w serwisie,

λ – współczynniki równoczesności obsługi żądań,

G_i – wektor adresów obiektów nie pobranych jeszcze przez klienta w ramach strony, której dotyczy i-te żądanie, $G_i = \left[u_{i1},...,u_{il},...,u_{iL}\right]$,

u_{il} – składowa wektora G_i, adres obiektu jeszcze nie pobranego ze strony, $l = 1,...,L$,

$\hat{T}_i^{s''}$ – wektor szacowanych czasów odpowiedzi na żądania dotyczące wskazanych w G_i obiektów dla s''-tego serwisu lokalnego, $\hat{T}_i^{s''} = \left[\hat{t}_{i1}''^{s''},...,\hat{t}_{il}''^{s''},...,\hat{t}_{iL}''^{s''}\right]$,

$\hat{t}_{il}''^{s''}$ – element wektora $\hat{T}_i^{s''}$, szacowany czas odpowiedzi dla l-tego obiektu i s''-tego serwisu lokalnego.

Zadanie projektowe 1: należy opracować projekt serwera pośredniczącego GGARDiB, który dla każdego przychodzącego żądania x_i wyznaczy na podstawie

wiedzy o obciążeniach $O_i''^1,...,O_i''^{S''}$ oraz wiedzy o przeszłych czasach odpowiedzi na żądania $\tilde{t}_1'',...,\tilde{t}_{i-1}''$ serwis lokalny w_i'' spośród S'' serwisów, który obsłuży żądanie x_i, tak, że $w_i'':\hat{t}_i''^{w_i''}=\min_{s''}\left\{\hat{t}_i''^{s''}:s''=1,...,S''\right\}$ oraz wyznaczy termin db_i ujęty w zmodyfikowanym żądaniu x_i' tak, aby spełniony był warunek $\forall_{p_i,i=1,2,...}\dot{t}_{p_i}\le t_{\max}$ przy założeniach, że $\neg\exists_{p_i}\left[\left(\dot{t}_{p_i}>t_{\max}\right)\wedge\left(\forall_i\forall_{s''\in\{1,...,S''\}}arl_i^{s''}=ars_{\max}^{s''}\right)\right]$ oraz $\forall_{x_i,s''\in\{1,...,S''\}}\left(arl_i^{s''}<arl_{\max}^{s''}\right)$.

Zadanie projektowe 2: należy opracować projekt przełącznika webowego GGARDiB, w którym decyzje podejmowane są w dwóch etapach. W pierwszym etapie dla każdego nadchodzącego żądania x_i', gdzie $i=1,2,...$, w przypadku, gdy $ars^{s''}(\tau_i^{(2)})=ars_{\max}^{s''}$, należy wyznaczyć uszeregowanie dla kolejki żądań HTTP Q_i działającej zgodnie z polityką EDF, w której wybór kolejności żądań realizowany jest na podstawie terminów ds_i, gdzie termin ds_i jest tak wyznaczany, aby nie został przekroczony termin db_i. W przypadku, gdy $ars^{s''}(\tau_i^{(2)})<ars_{\max}^{s''}$, żądanie x_i nie jest umieszczane w kolejce Q_i.

W drugim etapie dla każdego opuszczającego kolejkę żądania x_i lub żądania, które w kolejce nie było umieszczone, w oparciu o wiedzę o obciążeniach serwerów webowych $O_i^1,...,O_i^{S^{s''}}$ oraz wiedzę o przeszłych czasach odpowiedzi na żądania $\tilde{t}_1,...,\tilde{t}_{i-1}$, wyznaczyć serwer webowy w_i spośród $S^{s''}$ serwerów, dla którego spełniony jest warunek $\hat{t}_i^{w_i}\le\Delta ds_i$ o ile $\exists_{s\in\{1,...,S\}}\hat{t}_i^{s}\le\Delta ds_i$ i $ars^{s''}(\tau_i^{(2)})=ars_{\max}^{s''}$ lub w przeciwnym wypadku $w_i:\hat{t}_i^{w_i}=\min_s\left\{\hat{t}_i^{s}:s=1,...,S^{s''}\right\}$.

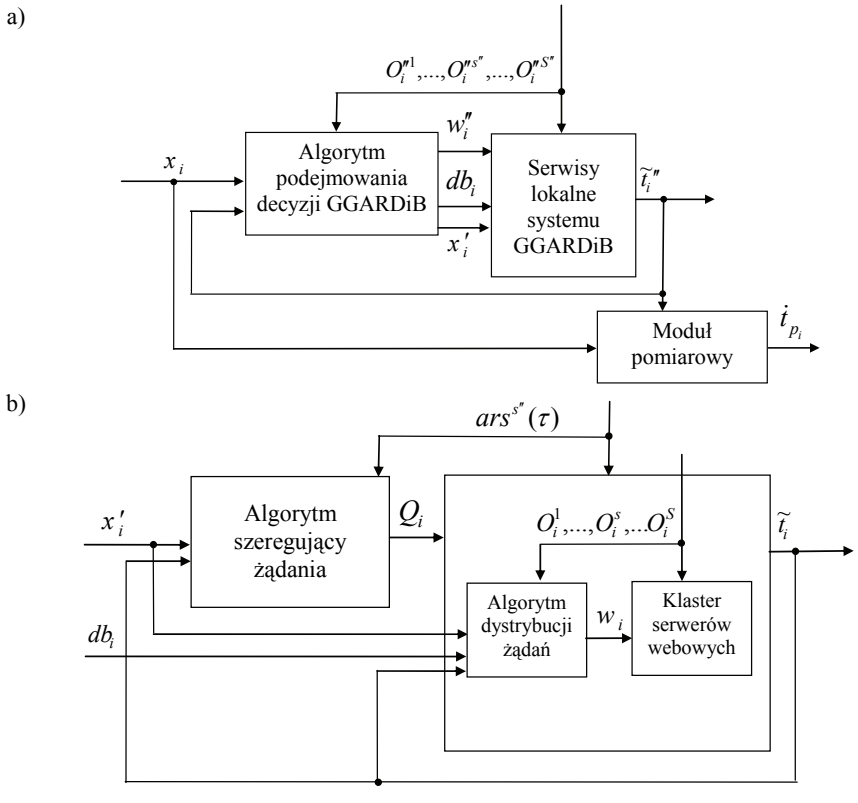
Podsumowując, proces podejmowania decyzji w systemie GGARDiB przebiega w trzech etapach. W pierwszym etapie serwer pośredniczący wybiera do obsługi żądania serwis lokalny, który oferuje najkrótszy czas odpowiedzi na żądanie. Równocześnie serwer pośredniczący wyznacza termin db_i, gdy obsługa żądania w serwisie lokalnym powinna się zakończyć. Termin ten jest tak wyznaczony, aby czas odpowiedzi dla całej strony nie przekroczył czasu $t_{\max}$. W drugim etapie przełącznik webowy w serwisie lokalnym szereguje żądania tak, by zachowany był termin db_i. W trzecim etapie wyznaczany jest do obsługi żądania serwer webowy, oferujący czas obsługi krótszy niż pozostały do terminu db_i czas.

Na rysunku 4.17 przedstawiony został ogólny schemat procesu podejmowania decyzji w serwerze pośredniczącym GGARDiB (rys. 4.17a) oraz przełączniku webowym GGARDiB (rys. 4.17b). Algorytm podejmowania decyzji w serwerze pośredniczącym wyznacza serwis lokalny w_i'' do obsługi żądania oraz termin db_i. Algorytm szeregujący żądania w przełączniku webowym wyznacza kolejkę żądań Q_i, natomiast algorytm dystrybucji żądań wyznacza serwer w_i, który obsłuży żądanie.

Jak wynika z prezentowanych założeń i opisu decyzje w systemie GGARDiB podejmowane są na poziomie globalnym w serwerze pośredniczącym (etap pierwszy procesu podejmowania decyzji) oraz na poziomie lokalnym w serwisie lokalnym (etap drugi i trzeci). Na poziomie globalnym serwer pośredniczący GGARDiB, podobnie jak serwer pośredniczący GARDiB, podejmuje decyzje tak, aby podnosić jakość obsługi, a nie zapewniać jakości dla całych stron WWW. Dopiero przełącznik webowy działający na poziomie lokalnym podejmuje tak decyzje, aby zapewniać czasy obsługi całych stron WWW nie dłuższe niż żądana wartość. Takie podejście ma znaczące konsekwencje i może obniżyć jakość działania systemu GGARDiB oraz ograniczyć pulę zastosowań systemu (zagadnienia te będą dalej rozpatrywane w rozdz. 4.3.4).

Trudności w opracowaniu efektywnych algorytmów dystrybucji żądań, które już na poziomie globalnym umożliwiałyby zapewnianie jakości usług związane są z tym, że decyzje podejmowane są w tym samym czasie w wielu różnych urządzeniach, które mogą przyjąć różne strategie postępowania odpowiednie do obsługi swoich klientów i równocześnie negatywnie wpływające na jakość obsługi w innych urządzeniach. Dodatkowo duże znaczenie ma to, że poszczególne serwisy lokalne rozmieszczone są w różnych lokalizacjach geograficznych i różnych częściach infrastruktury sieci rozległej. Przyjmując dotychczasową strategię pozostawiania nieobciążonych tych realizatorów (w tym wypadku serwisów lokalnych), dla których czas odpowiedzi na żądania jest krótki, należałoby wysyłać żądania do serwisów lokalnych nie tych najbliższych klientowi, lecz dalszych. W konsekwencji mogłoby dość do sytuacji, w której większość żądań klientów byłaby przekierowywana do serwisów lokalnych odległych od klienta. Takie postępowanie nie tylko zwiększałoby znacząco średni czas odpowiedzi dla wszystkich stron WWW, ale równocześnie niepotrzebnie podnosiłoby poziom ruchu w Internecie, zwiększając dyskomfort pracy innych użytkowników Internetu.

Opracowanie systemu, który już na poziomie globalnym umożliwiałby efektywną dystrybucję, w ramach której brany jest pod uwagę zadany czas odpowiedzi dla strony wymaga jeszcze dalszych prac.

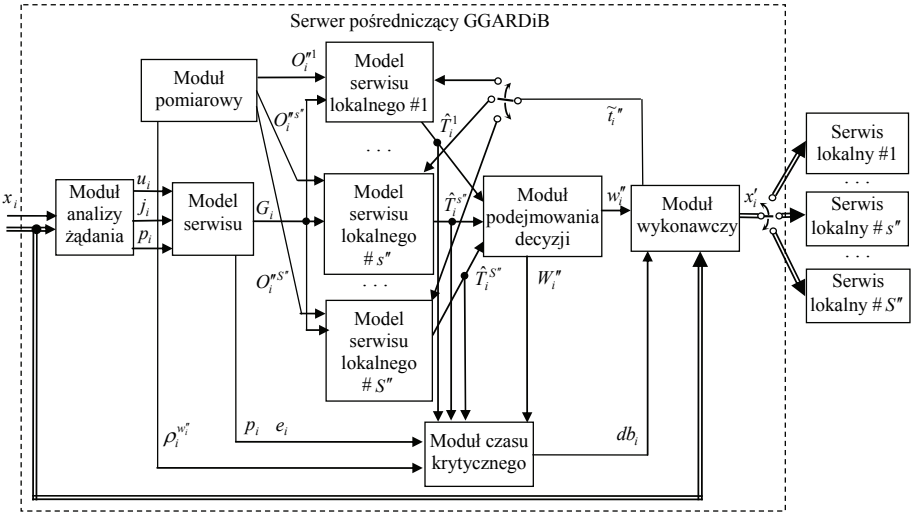


Rys. 4.17. Schemat prezentujący proces podejmowania decyzji w systemie GGARDiB: a) z udziałem serwera pośredniczącego, b) z udziałem przełącznika webowego

4.3.2. SERWER POŚREDNICZĄCY GGARDiB

Serwery pośredniczące zlokalizowane są w Internecie blisko potencjalnych klientów. Pojedynczy pośrednik diagnozuje stan sieci komputerowej na kierunku od pośrednika do serwisów lokalnych oraz stany serwisów lokalnych. Na podstawie otrzymanych danych szacowany jest czas odpowiedzi na żądanie HTTP dla wszystkich serwisów lokalnych, a następnie wyznaczany jest serwis lokalny, dla którego oszacowany czas odpowiedzi na żądanie będzie najmniejszy. Pośrednik przesyłając żądanie do serwisu lokalnego dodaje do żądania informacje o wymaganym czasie zakończenia obsługi żądania przez serwis.

Główne elementy serwera pośredniczącego to: moduł analizy żądania, moduł pomiarowy, model serwisu, moduł czasu krytycznego, modele serwisów lokalnych, moduł podejmowania decyzji, moduł wykonawczy. Ogólny schemat serwera pośredniczącego GGARDiB przedstawiony został na rys. 4.18.



Rys. 4.18. Schemat serwera pośredniczącego GGARDiB

Moduł analizy żądania działa w taki sam sposób, jak zostało to opisane w rozdz. 3.1.1 przy opisie metody WEDF. Po odebraniu żądania x_i przeprowadzana jest jego analiza i pobierane są: adres żądanego obiektu u_i, identyfikator użytkownika j_i, identyfikator strony p_i, w ramach której dany obiekt jest żądany. Identyfikatory strony i użytkownika zawarte są w informacjach w *cookie* ck_i żądania x_i.

Model serwisu w pośredniku ma podobną konstrukcję do modelu serwisu w serwerze szeregującym WEDF, mianowicie przechowuje informacje o obiektach HTTP udostępnianych w serwisie, ich wielkości, rodzaju oraz przynależności poszczególnych obiektów do stron. Na wejściu do modelu serwisu przekazywane są informacje o adresie żądanego obiektu u_i, identyfikatorze użytkownika j_i i identyfikatorze strony p_i.

Model serwisu przekazuje informacje $G_i = \left[u_{i1},...,u_{il},...,u_{iL}\right]$ o adresie obiektu żądanego w x_i oraz adresach obiektów należących do strony, ale nie żądanych jeszcze przez klienta w ramach danej strony, u_{i1} jest adresem żądanego obiektu i $u_{i1} = u_i$, natomiast $u_{i2},...,u_{il},...,u_{iL}$ są adresami obiektów jeszcze nie żądanych. Dodatkowo przekazywana jest również wartość czasu tp_{p_i}. Czas ten mierzony jest od momentu przyjścia pierwszego żądania, wysłanego przez klienta dotyczącego obiektu danej strony, do momentu nadejścia żądania x_i.

Serwer pośredniczący zawiera S'' **modeli serwisów lokalnych**, czyli tyle, ile serwisów lokalnych znajduje się w całym serwisie. Każdy model serwisu lokalnego jest modelem systemu obsługi żądań HTTP opisanym w rozdz. 2.1, przy czym na wejściu modelu podawany jest wektor G_i adresów obiektów, a na wyjściu uzyskiwany jest wektor $\hat{T}_i^{s''} = \left[\hat{t}_{i1}''^{s''},...,\hat{t}_{il}''^{s''},...,\hat{t}_{iL}''^{s''}\right]$ szacowanych czasów odpowiedzi na żądania dotyczące wskazanych obiektów, gdzie $\hat{t}_{i1}''^{s''}$ (oznaczony również jako $\hat{t}_i''^{s''}$) jest szacowanym czasem odpowiedzi na żądanie x_i, $\hat{t}_{i2}''^{s''},...,\hat{t}_{iL}''^{s''}$ są szacowanymi czasami odpowiedzi dla pozostałych obiektów. Modele serwisów lokalnych wyznaczają poszczególne wartości $\hat{t}_{il}''^{s''}$, $l = 1,...,L$, szeregowo. Do szacowania czasów wykorzystywane są informacje $O_i''^{s''}$, $s'' = 1,...,S''$, o stanie obciążenia serwisów lokalnych i sieci na kierunkach do serwisów lokalnych.

Moduł pomiarowy pozyskuje informacje $O_i''^1,...,O_i''^{s''},...,O_i''^{S''}$, gdzie $O_i''^{s''} = \left[a_i^{s''}, b_i^{s''}\right]$ i $a_i^{s''} = arl_i^{s''}$ i $b_i^{s''} = btt_i^{s''}$. Wartość $arl_i^{s''}$ jest liczbą żądań równocześnie obsługiwanych w s''-tym serwisie lokalnym. Wartość $btt_i^{s''}$ jest czasem transferu próbnego obiektu od s''-tego serwisu lokalnego do pośrednika. Obie miary obciążenia są pozyskiwane przez moduł pomiarowy w taki sam sposób, jak zostało to opisane w przypadku metody GARDiB. Dodatkowo moduł pomiarowy przekazuje informacje $\rho_i^{w_i''}$, dotyczące efektywnej przepustowości na łączu od serwisu lokalnego, który został wybrany do obsługi żądania x_i, do serwera pośredniczącego.

Informacje o oszacowanych czasach odpowiedzi $\hat{T}_i^1,...,\hat{T}_i^{s''},...,\hat{T}_i^{S''}$ przekazywane są z modeli serwisów lokalnych do modułu podejmowania decyzji. **Moduł podejmowania decyzji** wyznacza wektor $W_i = \left[w_{i1}'',...,w_{il}'',...,w_{iL}''\right]$, gdzie w_{i1}'' jest numerem serwisu lokalnego wybranym do obsługi żądania x_i, $w_{i2}'',...,w_{il}'',...,w_{iL}''$ są numerami serwisów lokalnych, które mogłyby obsłużyć pozostałe żądania, dotyczące nie pobranych jeszcze obiektów strony, przy czym $w_{il}'' \in \{1,...,S''\}$. Decyzje w_{il}'' wyznacza się zgodnie z zależnością:

$$w_{il}'' : \hat{t}_{il}''^{w_{il}''} = \min\left\{\hat{t}_{il}''^{s''} : s'' = 1,...,S''\right\}. \tag{4.8}$$

Decyzja w_{i1}'' dotyczy żądania x_i i jest również oznaczona w_i''.

W **module czasu krytycznego** wyznaczany jest termin db_i, w którym serwis lokalny powinien zakończyć obsługę żądania x_i i zacząć przesyłanie żądanego obiektu do pośrednika. Dokładniej rzecz biorąc db_i jest terminem, w którym przełącznik webowy powinien otrzymać w całości żądany obiekt.

Termin db_i wyznaczany jest zgodnie ze wzorem $db_i = \Delta db_i + \tau_i^{(1)} - h_i^{s''}\Big|_{s''=w_i''}$, gdzie $\tau_i^{(1)}$ jest momentem, gdy nadeszło żądanie x_i. Czas Δdb_i jest maksymalnym czasem odpowiedzi na pojedyncze żądanie, który nie powinien być przekroczony tak, aby czas odpowiedzi dla całej strony nie był dłuższy niż $t_{\max}$. Czas $h_i^{s''}$ jest czasem transferu odpowiedzi na i-te żądanie z wybranego do obsługi serwisu lokalnego do serwera pośredniczącego. Wartość $h_i^{s''}$ obliczana jest z zależności $h_i^{s''} = \nu_i / \rho_i^{s''}$, gdzie ν_i jest wielkością odpowiedzi na i-te żądanie. Wartość Δdb_i jest wyznaczana dla wybranego serwisu w_{i1}'' zgodnie zależnością:

$$\Delta db_i = \hat{t}_{i1}^{\prime\prime w_{i1}''} \frac{(t_{\max} - tp_{p_i})}{\lambda \sum_{l=1}^{L} \hat{t}_{il}^{\prime\prime w_{il}''}}, \tag{4.9}$$

gdzie: λ jest współczynnikiem równoczesności obsługi żądań wyznaczanym w sposób eksperymentalny.

Może się zdarzyć, że Δdb_i przyjmie wartość ujemną, jeżeli $t_{\max} - tp_{p_i} < 0$.

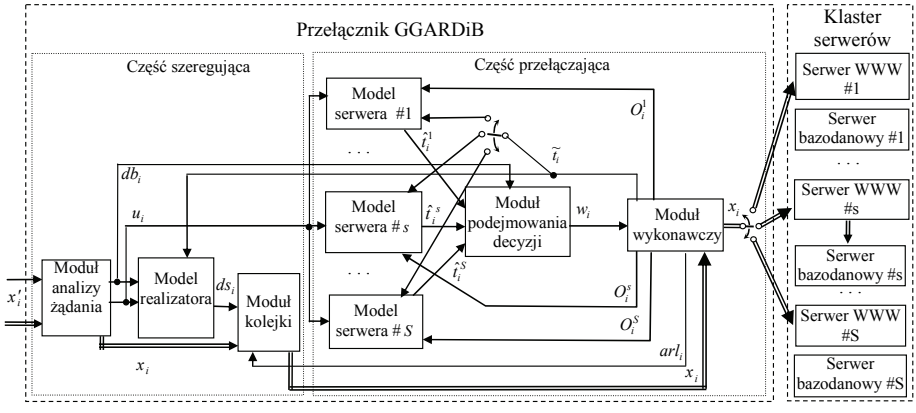
Zadaniem **modułu wykonawczego** jest przekierowanie do wybranego serwisu lokalnego w_i'' zmodyfikowanego żądania x_i'. Żądanie x_i' powstaje z oryginalnego żądania x_i, po dodaniu w nagłówku żądania w polu *cookie* informacji o terminie db_i, w ten sposób, że $x_i' = \langle u_i, ck_i' \rangle$, gdzie $ck_i' = ck_i \cup \{db_i\}$.

Tak zmodyfikowane żądanie przesyłane jest do serwisu lokalnego, a precyzyjniej do przełącznika GGARDiB. W czasie, gdy wybrany serwis lokalny obsługuje żądanie x_i', moduł wykonawczy mierzy rzeczywisty czas $\tilde{t}_i''$ odpowiedzi na żądanie. W momencie, gdy żądany obiekt dotrze w całości do pośrednika, moduł wykonawczy przesyła informacje o czasie odpowiedzi do modelu serwisu lokalnego odpowiadającego serwisowi, który obsłużył żądanie. Model serwisu lokalnego aktualizuje w procesie adaptacji wiedzę o serwisie lokalnym.

4.3.3. PRZEŁĄCZNIK WEBOWY GGARDiB

Drugim, po serwerze pośredniczącym, najważniejszym elementem systemu jest przełącznik GGARDiB, przełącznik jest częścią serwisu lokalnego. Przełącznik otrzymuje żądania HTTP od serwera pośredniczącego. Zadanie przełącznika polega na dystrybucji żądań w lokalnym klastrze serwerów webowych, a precyzyjniej, na wybraniu odpowiedniego serwera do obsługi żądania,

przesłaniu żądania do serwera, odebraniu od serwera obiektu HTTP i przesłaniu go do serwera pośredniczącego. Schemat przełącznika GGARDiB przedstawiony został na rys. 4.19.



Rys. 4.19. Schemat przełącznika GGARDiB

Przełącznik GGARDiB składa się z części szeregującej oraz przełączającej. W skład części szeregującej wchodzą: moduł analizy żądania, model realizatora, moduł kolejki. W skład części przełączającej wchodzą: moduły modeli serwerów, moduł podejmowania decyzji i moduł wykonawczy. W dalszym opisie działania przełącznika GGARDiB pominięty zostanie indeks s'' wskazujący na numer serwisu lokalnego obsługującego żądanie.

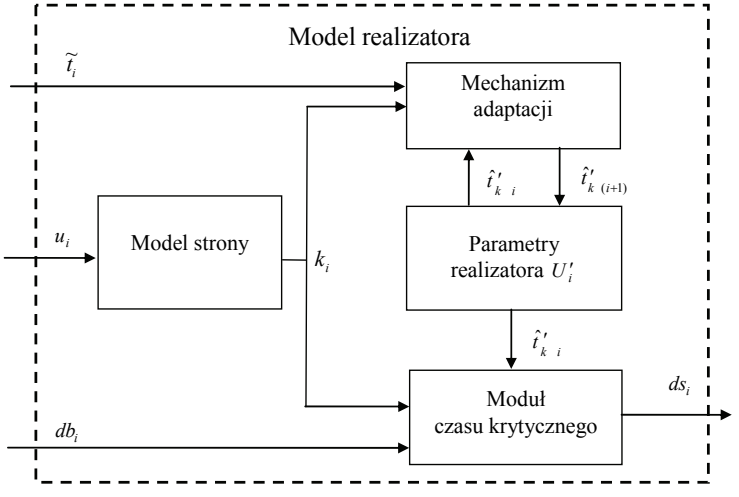
Przełącznik po otrzymaniu żądania $x_i' = \langle u_i, ck_i' \rangle$ przekazuje je do modułu analizy żądania.

Moduł analizy żądania pobiera z żądania następujące informacje: adres żądanego obiektu u_i, a z pola *cookie* ck_i' termin db_i, w którym serwis lokalny powinien zakończyć obsługę żądania. Równocześnie z pola *cookie* usuwane są informacje dotyczące terminu db_i, w wyniku czego uzyskiwane jest oryginalne żądanie x_i przekazane dalej do modułu kolejki.

Model realizatora w przełączniku GGARDiB wyznacza termin ds_i wskazujący na moment, w którym powinna rozpocząć się obsługa i-tego żądania przez serwer webowy. Model realizatora zawiera następujące moduły: model strony, mechanizm adaptacji, parametry realizatora i moduł czasu krytycznego.

Model strony wyznacza klasę k_i żądanego obiektu na podstawie adresu obiektu u_i.

Moduł mechanizmu adaptacji oraz moduł parametrów realizatora $U_i' = \left[\hat{t}_{1i}', ..., \hat{t}_{Ki}\right]$, działają w taki sam sposób, jak zostało to opisane w rozdz. 4.1.2, przy opisie systemu WEDF.



Rys. 4.20. Schemat modelu realizatora w przełączniku MLF

Moduł czasu krytycznego wyznacza termin ds_i na podstawie zależności:

$$ds_i = db_i - \hat{t}'_{ki}, \tag{4.10}$$

gdzie $k = k_i$, $\hat{t}'_{ki}$ jest elementem wektora parametrów realizatora oraz jest szacowanym czasem odpowiedzi na żądanie należące do k-tej klasy przy obciążeniu realizatora $arl_{\max}$.

Moduł kolejki w przełączniku GGARDiB działa podobnie jak moduł kolejki w przełączniku MLF. Żądania w module są uszeregowane w kolejce zgodnie z terminami ds_i, $i = 1,2,...$. Moduł kolejki przyjmuje żądania do kolejki, jeżeli $ars(\tau_i^{(2)}) = ars_{\max}$, w przeciwnym przypadku żądania przekazywane są bezpośrednio do modułu wykonawczego. Żądanie x_i opuszcza kolejkę Q_i, wtedy gdy wszystkie żądania umieszczone w kolejce przed nim opuszczą kolejkę oraz gdy wystąpi zdarzenie w pewnym momencie $\tau_i^{(3)}$ takie, że spełniony zostanie warunek $ars(\tau_i^{(3)}) < ars_{\max}$.

Część przełączająca przełącznika GGARDiB składa się z modeli serwerów, modułu podejmowania decyzji oraz modułu wykonawczego. Do części przełączającej przekazywany jest adres żądania u_i, żądanie x_i oraz termin db_i zakończenia obsługi żądania przez serwer webowy. Budowa i sposób działania części przełączającej przełącznika GGARDiB jest taka sama jak budowa i działanie części przełączającej przełącznika MLF i nie będzie w tym rozdziale dokładnie opisywana.

4.3.4. BADANIA DOTYCZĄCE SYSTEMU GGARDiB

Przeprowadzone zostały badania symulacyjne mające na celu określenie, czy system webowy GGARDiB pracuje w taki sposób, aby utrzymywać wartość miary satysfakcji użytkowników na poziomie wyższym niż w przypadku dotychczas stosowanych i badanych systemów webowych globalnie rozproszonych. Program symulacyjny składał się podobnie jak w przypadku badań nad systemem GARDiB z następujących modułów: modułu generatora żądań, modułu serwera pośredniczącego, modułu Internetu, modułu serwisu lokalnego. Schemat przyjętego modelu symulacyjnego jest taki sam jak dla systemu GARDiB i został przedstawiony na rys. 3.12, jednak działanie poszczególnych modułów jest inne.

Moduł generatora żądań skonstruowany został w taki sposób, jak moduł generatora żądań opisany w rozdz. 4.1.4 dla systemu WEDF. Moduł Internetu został skonstruowany podobnie jak w programie symulacyjnym dla systemu GARDiB. Moduł serwera pośredniczącego umożliwiał dystrybucję żądań zgodnie z metodą GGARDiB i GARDiB oraz algorytmami: Round-Robin (RR), Weighted Round-Robin Load (WRR_L) i Weighted Round-Robin Transfer (WRR_T).

Moduł serwisu lokalnego skonstruowany był podobnie jak w programach symulacyjnych opracowanych dla metod GARD i GARDiB, przy czym możliwości modułu przełącznika webowego zostały tak zmienione, by prócz obsługi takich algorytmów dystrybucji żądań jak LFNRD, LARD, CAP, WRR, RR przełącznik mógł również dystrybuować żądania zgodnie z metodą GGARDiB.

W badaniach symulacyjnych, w których serwer pośredniczący pracował zgodnie z metodą GGARDiB, przełącznik webowy również pracował zgodnie z tą metodą. W przypadku innych metod dystrybucji żądań zastosowanych w pośredniku przełącznik sieciowy wykorzystywał algorytm LFNRD.

Przeprowadzone badania wstępne wykazały, że dla systemu GGARDiB uzyskiwane są największe wartości satysfakcji, jeżeli czas transferu odpowiedzi na żądanie HTTP jest nie większy niż o jeden rząd od czasu odpowiedzi żądania HTTP w serwisie lokalnym. Jest to związane z tym, że na termin, gdy żądany obiekt znajdzie się w całości w serwerze pośredniczącym, ma duży wpływ czas transferu obiektu z przełącznika do pośrednika. Obecnie stosowane sieci rozległe oferują dość małe transfery, dlatego też czasy przesłania obiektów HTTP przez sieć rozległą są często znacznie dłuższe niż czasy obsługi w serwisach lokalnych. Dodatkowo transfer między dwoma punktami w sieci często charakteryzuje się znaczną zmiennością i bywa, że różnica między zmierzonym a obliczonym czasem transferu obiektu może być większa niż czas obsługi żądania w serwisie lokalnym. W takim wypadku wartości satysfakcji nie będą wyższe w serwisach stosujących metodę GGARDiB niż serwisach stosujących metodę GARDiB.

Ze względu na wskazania dotyczące zastosowań systemu GGARDiB przyjęto, że w module Internetu wszystkie czasy przesyłania danych zostały po-

dzielone przez określony współczynnik tak, aby czasy te nie były znacząco dłuższe od czasów obsługi żądań w serwisach lokalnych. Przyjęta wartość współczynnika wynosiła 30.

W ramach eksperymentów, których wyniki prezentowane są w tym opracowaniu, przeprowadzone zostały badania symulacyjne dla pięciu różnych wariantów konfiguracji systemu webowego. W pierwszym wariancie wykorzystany został system webowy zawierający trzy serwisy lokalne, z których każdy składał się z trzech serwerów WWW i bazodanowych. Każdy z trzech serwisów lokalnych został obciążony żądaniami pochodzącymi od 25% użytkowników. Również na pośrednika obserwowanego w doświadczeniach zostało przekierowane 25% żądań użytkowników. Badania przeprowadzone zostały dla trzech serwisów lokalnych umiejscowionych w Holandii, USA i Australii. Żądane czasy odpowiedzi dla całych stron wynosiły odpowiednio $t^s_{\max} = 4000\,\text{ms}$ i $t^h_{\max} = 8000\,\text{ms}$. Wariant pierwszy oznaczony został NL/3/25%; USA/3/25%; AU/3/25%; pośrednik/25%; $t^s_{\max} = 4000\,\text{ms}$; $t^h_{\max} = 8000\,\text{ms}$.

W wariancie drugim wykorzystana została taka sama konfiguracja serwisów lokalnych, i obciążeń. Zmienione zostały natomiast żądane czasy odpowiedzi dla stron na $t^s_{\max} = 2000\,\text{ms}$ i $t^h_{\max} = 4000\,\text{ms}$. Wariant ten oznaczony został NL/3/25%; USA/3/25%; AU/3/25%; pośrednik/25%; $t^s_{\max} = 2000\,\text{ms}$; $t^h_{\max} = 4000\,\text{ms}$. W wariancie trzecim konfiguracja serwisów lokalnych była taka sama jak w dwóch opisanych wyżej eksperymentach, przy czym obciążenie serwisów lokalnych było inne. Na serwer w Holandii wysłane zostało 30% użytkowników, do USA 25%, do Australii 20% na obserwowanego pośrednika 25% użytkowników. Żądane czasy odpowiedzi dla stron wynosiły $t^s_{\max} = 4000\,\text{ms}$ i $t^h_{\max} = 8000\,\text{ms}$. Wariant ten oznaczony został NL/3/30%; USA/3/25%; AU/3/20%; pośrednik/25%; $t^s_{\max} = 4000\,\text{ms}$; $t^h_{\max} = 8000\,\text{ms}$. W kolejnym wariancie wykorzystanych zostało pięć serwisów lokalnych, umiejscowionych w Holandii, USA, Australii, Brazylii i Polsce. Każdy z serwisów lokalnych zawierał po trzy serwery WWW i bazodanowe, na wszystkie serwisy lokalne oraz pośrednika wysłanych zostało po 16,666% żądań użytkowników. Żądane czasy odpowiedzi dla stron wynosiły $t^s_{\max} = 4000\,\text{ms}$ i $t^h_{\max} = 8000\,\text{ms}$. Wariant oznaczony został NL/3/16,666%; USA/3/16,666%; AU/3/16,666%; BR/3/16,666%; PL/3/16,666%; pośrednik/16,666%; $t^s_{\max} = 4000\,\text{ms}$; $t^h_{\max} = 8000\,\text{ms}$. W ostatnim z opisywanych wariantów wykorzystane zostały trzy różne serwisy lokalne, przy czym serwis w Holandii zawierał tylko jeden zestaw serwera WWW i bazodanowego i był obciążony 10% żądań, serwis w USA zawierał dwa zestawy serwerów i był obciążony 20% żądań, natomiast serwis w Australii zawierał trzy zestawy serwerów i był obciążony 30% żądań, pośrednik otrzymywał 40% żądań. Żądane czasy odpowiedzi dla stron wynosiły

podobnie jak poprzednio $t^s_{\max} = 4000\,\text{ms}$; $t^h_{\max} = 8000\,\text{ms}$. Wariant oznaczony został NL/1/10%; USA/2/20%; AU/3/30%; pośrednik/40% $t^s_{\max} = 4000\,\text{ms}$; $t^h_{\max} = 8000\,\text{ms}$.

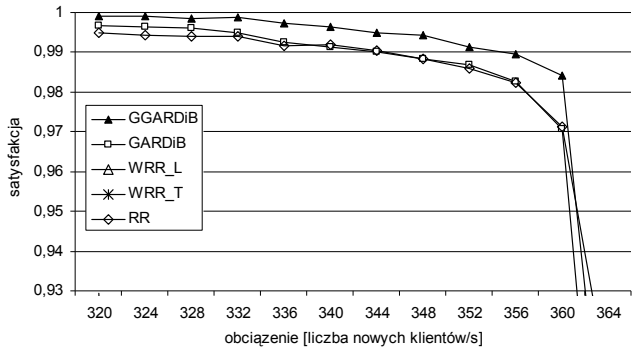
W trakcie badań mierzona była średnia wartość satysfakcji, czas odpowiedzi dla stron oraz średnie czasy odpowiedzi na żądania. Na rys. 4.21 przedstawione zostały wykresy średniej wartości satysfakcji w funkcji obciążenia (liczby nowych klientów tworzonych w ciągu sekundy) dla trzech pierwszych konfiguracji serwisów. Na rys. 4.22 przedstawione zostały wykresy poziomu satysfakcji w funkcji obciążenia dla systemu webowego z heterogenicznymi serwisami lokalnymi oraz pięcioma serwisami lokalnymi. Rys 4.23a przedstawia średnie czasy odpowiedzi na żądania oraz rys. 4.23b dystrybuanty czasów odpowiedzi dla stron dla poszczególnych metod dystrybucji żądań.

Otrzymane wyniki badań symulacyjnych wskazują, że dla większości przeprowadzonych eksperymentów średnia wartość satysfakcji dla metody GGARDiB jest najwyższa, zarówno dla małych obciążeń serwisów jak i dużych graniczących z obciążeniem, powyżej którego serwis nie jest już w stanie obsłużyć wszystkich otrzymanych żądań HTTP. Wysokie wartości satysfakcji otrzymano zarówno dla krótszych jak i dłuższych czasów $t^s_{\max}$ i $t^h_{\max}$ oraz różnych obciążeń serwisów lokalnych. Z wyników badań wnioskować można, że metoda GGARDiB może być stosowana dla serwisów zawierających niewielką liczbę serwisów lokalnych jak i tych trochę większych. Dość dobre wyniki otrzymane zostały dla serwisu zawierającego serwisy lokalne o różnych mocach obliczeniowych – rys. 4.22b.

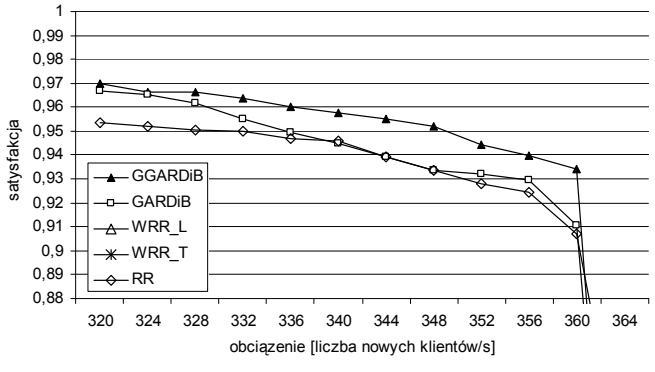
Na większości prezentowanych wykresów na rys. 4.21 i 4.22 średnie wartości satysfakcji dla algorytmów WRR_L i WRR_T są na tyle małe, że nie zostały ujęte na wykresie. Przy czym, o ile w przypadku algorytmu WRR_T przy większym obciążeniu wartość satysfakcji zbliżała się do $0{,}1$, to dla algorytmu WRR_L wartość ta wahała się w przedziale od 0,8 do 0,7. Dość dobre wyniki otrzymane zostały dla algorytmu dystrybucji RR w badaniach, w których serwisy lokalne były obciążone w sposób równomierny. Algorytm ten jednak nie powinien być stosowany w rozwiązaniach przemysłowych, ponieważ w praktyce bardzo rzadko zdarza się, by serwisy lokalne były obciążone równomiernie.

Na rys. 4.23 przedstawione zostały średnie czasy odpowiedzi na żądania oraz dystrybuanty czasów odpowiedzi dla całych stron. Z wykresów na rys. 4.23a wynika, że średnie czasy odpowiedzi dla żądań HTTP są dla metody GGARDiB dłuższe niż dla RR i GARDiB. Należy jednak zwrócić uwagę, że w tych samych badaniach średnie wartości satysfakcji są wyższe dla GGARDiB niż dla pozostałych metod dystrybucji żądań. Równocześnie, analizując dystrybuanty czasów odpowiedzi dla stron widać, że dystrybuanta dla metody GGARDiB najszybciej dąży do wartości 1, a co za tym idzie znaczna większość użytkowników będzie oczekiwała krócej na otrzymanie całych stron.

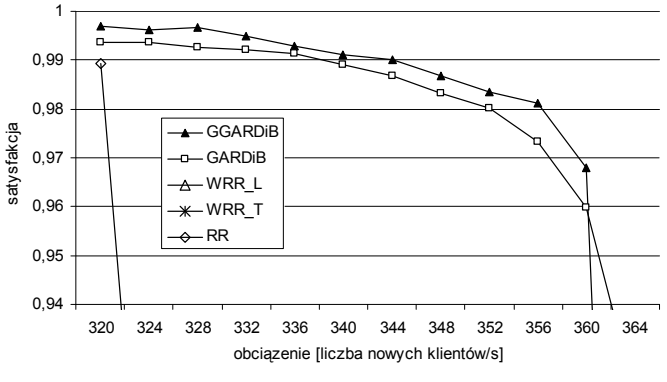
a)



b)

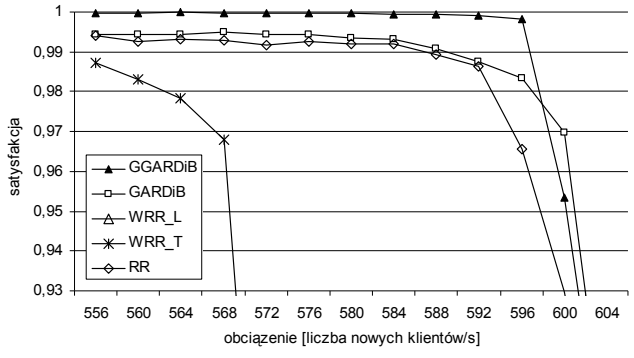


c)

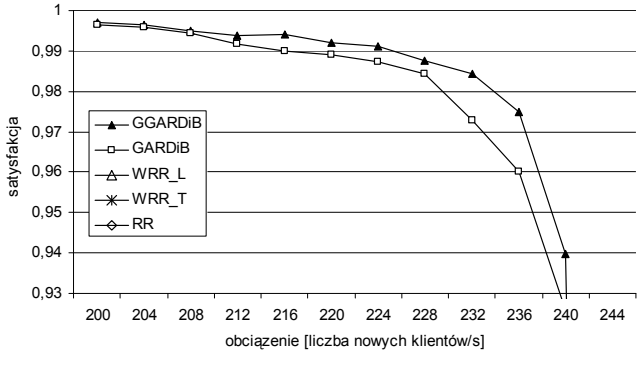


Rys. 4.21. Średnia wartość satysfakcji w funkcji obciążenia dla wariantów:
a) NL/3/25%; USA/3/25%; AU/3/25%; pośrednik/25%; $t^s_{\max} = 4000$ ms; $t^h_{\max} = 8000$ ms,
b) NL/3/25%; USA/3/25%; AU/3/25%; pośrednik/25%; $t^s_{\max} = 2000$ ms; $t^h_{\max} = 4000$ ms,
c) NL/3/30%; USA/3/25%; AU/3/20%; pośrednik/25%; $t^s_{\max} = 4000$ ms; $t^h_{\max} = 8000$ ms

a)

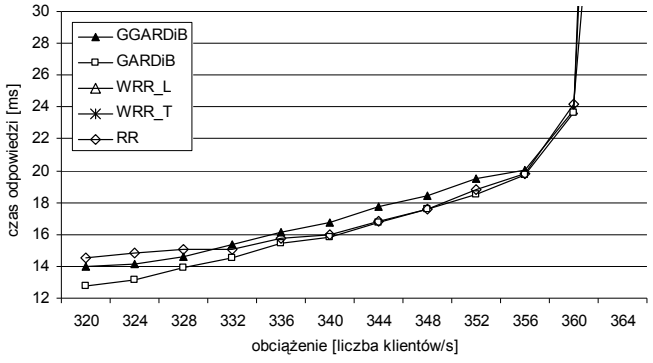


b)

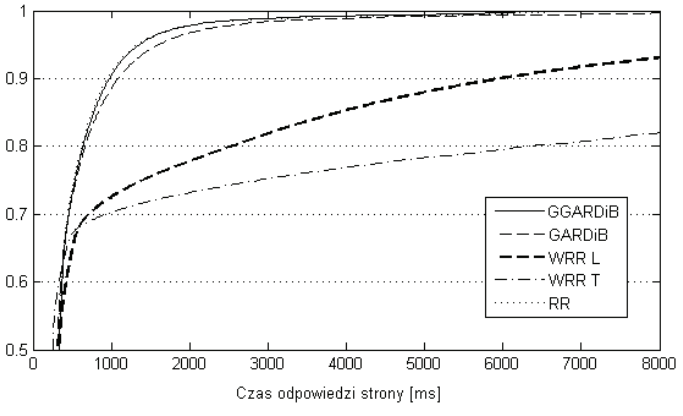


Rys. 4.22. Średnia wartość satysfakcji w funkcji obciążenia dla wariantów:
a) NL/3/16,666%; USA/3/16,666%; AU/3/16,666%; BR/3/16,666%; PL/3/16,666%; pośrednik/16,666%; $t^{s}_{\max} = 4000$ ms; $t^{h}_{\max} = 8000$ ms,
b) NL/1/10%; USA/2/20%; AU/3/30%; pośrednik/40% $t^{s}_{\max} = 4000$ ms; $t^{h}_{\max} = 8000$ ms

a)



b)



Rys. 4.23. Wyniki badań dla wariantu NL/3/25%; USA/3/25%; AU/3/25%; pośrednik/25%; $t^{s}_{\max} = 4000$ ms; $t^{h}_{\max} = 8000$ ms: a) średnie czasy odpowiedzi na żądanie w funkcji obciążenia, b) dystrybuanta czasów odpowiedzi dla strony przy obciążeniu 356 klientów/s

Na zakończenie przeprowadzone zostały badania oraz analizy mające na celu określenie wydajności serwera pośredniczącego oraz przełącznika webowego GGARDiB. Implementacja serwera pośredniczącego wykorzystanego w badaniach różniła się nieznacznie od sposobu pracy serwera pośredniczącego opisanego w rozdz. 4.3.2. Różnica polegała na tym, że w zaimplementowanym serwerze pośredniczącym decyzje o alokacji żądań dotyczących poszczególnych obiektów w serwisach lokalnych były podejmowane tylko raz dla danej strony i klienta w momencie przyjścia pierwszego żądania o obiekt strony. W momencie nadejścia żądań, dotyczących kolejnych obiektów strony wykorzystywane były decyzje dotyczące danego obiektu, ale podjęte już przy obsłudze pierwszego obiektu strony. Dlatego też średni czas podjęcia decyzji oraz wydajność w serwerze pośredniczącym GGARDiB podobne były jak dla serwera pośredniczącego GARDiB. Wydajność serwera pośredniczącego GGARDiB wynosiła 458 tys. żądań na sekundę, natomiast średni czas podjęcia decyzji wynosił 0,00218 ms. Badania przeprowadzone zostały dla serwera opisanego w rozdz.

3.1.3. Przy przyjętej implementacji złożoność obliczeniowa algorytmu podejmowania decyzji jest taka sama jak dla algorytmu w systemie GARDiB. Wydajność i średni czas podjęcia decyzji dla przełącznika webowego GGARDiB podobne były jak dla przełącznika MLF i wynosiły odpowiednio: wydajność 608 tysięcy żądań na sekundę, natomiast średni czas odpowiedzi na żądanie wynosił 0,001644 ms.

Złożoność obliczeniowa algorytmu w części szeregującej wynosi $O(\log N + arl)$, gdzie N jest liczbą rodzajów obiektów pobieranych w ramach serwisu, arl jest liczbą żądań równocześnie obsługiwanych. Złożoność obliczeniowa algorytmu w części przełączającej wynosi $O(S)$, podobnie jak dla przełącznika MLF, gdzie S jest liczbą serwerów WWW pracujących w klastrze. Z zaprezentowanych analiz i wyników dotyczących wydajności serwera pośredniczącego oraz przełącznika wynika, że obydwa urządzenia nie powinny stać się „wąskimi gardłami" systemu.

Pomimo że uzyskane zostały dość dobre wyniki dotyczące jakości działania systemu GGARDiB (zaprezentowane na rys. 4.21, 4.22 i 4.23 oraz w [219]), zastosowanie obecnie wskazanego systemu w sieci rozległej Internet nie przyniesie oczekiwanych rezultatów, tj. zapewniania czasów odpowiedzi dla stron WWW, ze względu na długie czasy przesyłania danych w sieci w stosunku do czasów odpowiedzi w serwisach lokalnych. Istnieje więc potrzeba opracowania metody globalnej dystrybucji żądań, w której na poziomie globalnym, a nie tylko lokalnym, realizowana byłaby usługa z zadanym poziomem jakości usług. Należy tu jednak zaznaczyć, że opracowanie metody GGARDiB wskazuje, że jest możliwe opracowanie wieloserwerowego systemu webowego realizującego obsługę żądań HTTP w taki sposób, aby zapewniać, by czasy odpowiedzi dla stron webowych były nie dłuższe od zadanej wartości.

4.4. PODSUMOWANIE

W rozdziale opisane zostały trzy systemy WEDF, MLF oraz GGARDiB umożliwiające szeregowanie i dystrybucje żądań HTTP w taki sposób, aby nie były przekraczane zadane czas odpowiedzi dla strony. Zaprezentowane propozycje stanowią przegląd rozwiązań umożliwiających gwarantowanie usług w małych webowych systemach jednoserwerowych, w systemach klastrowych lokalnie rozmieszczonych oraz systemach klastrowych globalnie rozmieszczonych.

Prezentowane autorskie rozwiązania otwierają nową grupę rozwiązań w zakresie systemów webowych z zadanym czasem odpowiedzi ocenianym z punktu widzenia użytkownika końcowego.

5. WNIOSKI KOŃCOWE

Przedmiotem monografii był problem projektowania systemów webowych z jakością usług. W rozdziale pierwszym monografii przedstawiona została klasyfikacja systemów webowych z jakością usług, w której wyróżnione zostały trzy grupy rozwiązań: najczęściej stosowane obecnie systemy z kryterium wydajności, dynamicznie rozwijająca się grupa systemów z kryterium czasowym oraz coraz częściej spotykane systemy z kryterium biznesowym. Celem monografii było przedstawienie opracowanych projektów systemów webowych z kryterium czasowym. W ramach monografii zaprezentowany został kompleks rozwiązań stosowanych w systemach z kryterium czasowym. Wskazane koncepcje systemów były rozwijane od początku powstania grupy systemów webowych z kryterium czasowym, stanowią prekursorskie rozwiązania w tym zakresie oraz tworzą trzon wskazanej grupy. W monografii prezentowane są rozwiązania należące do dwóch grup systemów webowych z kryterium czasowym: systemy minimalizujące czasy odpowiedzi oraz systemy z zadanym czasem odpowiedzi. Przedstawione w postaci projektów rozwiązania mogą stanowić podstawę w budowie systemów webowych różnej skali, począwszy od systemów zawierających jeden serwer webowy poprzez systemy kilkuserwerowe, kończąc na systemach zawierających kilkadziesiąt serwerów umieszczonych w różnych lokalizacjach.

Opracowanie systemów webowych, w których w procesie podejmowania decyzji brane są pod uwagę czasy odpowiedzi na żądania HTTP wymagało zaprojektowania odpowiedniego modelu serwisu webowego lub modelu podsystemów serwisu webowego. Opracowanie odpowiedniego uniwersalnego modelu umożliwiło przygotowanie rozwiązań podnoszących jakość obsługi oraz gwarantujących jakość. Dlatego też monografia podzielona została na następujące trzy części:

1. projekt uniwersalnego modelu rozmyto-neuronowego, umożliwiającego modelowanie elementów systemu webowego lub też całego systemu,
2. opracowania trzech projektów systemów webowych, minimalizujących czasy odpowiedzi na żądania HTTP,
3. opracowania trzech projektów systemów webowych z zadanym czasem odpowiedzi dla stron webowych.

Sześć zaprezentowanych systemów zostało tak opracowanych, aby ich wdrożenie nie wymagało modyfikacji protokołów sieciowych żadnej z siedmiu warstw modelu ISO/OSI. Dla każdego z systemów przedstawione zostały zadania projektowe oraz metody opisujące sposób działania systemu wraz z al-

gorytmami podejmowania decyzji, również zaprezentowane zostały projekty urządzeń lub oprogramowania, umożliwiające pracę systemów zgodnie ze wskazaną metodą. Wszystkie zaprezentowane algorytmy podejmowania decyzji umożliwiają pracę w czasie rzeczywistym. Proponowane w omawianych systemach algorytmy są algorytmami heurystycznymi, dlatego też do weryfikacji przedstawionych rozwiązań wykorzystane zostały metody symulacyjne. Dla każdego z przedstawionych systemów zbudowane zostało stanowisko laboratoryjne, którego głównymi elementami były programy symulacyjne, umożliwiające porównanie proponowanych rozwiązań z rozwiązaniami najczęściej stosowanymi oraz referencyjnymi. Dla każdego z omawianych systemów przeprowadzone zostały badania, a na podstawie ich wyników określić można jakość pracy proponowanych rozwiązań na tle rozwiązań już znanych oraz zakres zastosowań nowych metod. Wyniki badań symulacyjnych zostały szczegółowo przedyskutowane.

Materiał zaprezentowany w pierwszej części monografii (rozdz. 2) dotyczył projektu rozmyto-neuronowego modelu systemu obsługi żądań HTTP. Opisany uniwersalny model umożliwia modelowanie zarówno całych złożonych systemów webowych jak również jego podsystemów. Model ten cechuje się: możliwością pracy w czasie rzeczywistym, elastycznością w zastosowaniach, możliwością adaptacyjnego dostosowywania się do zmieniających się warunków, pracą w warunkach niepewności, gdy informacja o działaniu systemu webowego jest niedokładna. Ważną również cechą zastosowanego modelu jest fakt, że systemy go wykorzystujące nie wymagają złożonej wstępnej konfiguracji, a sam model dopasowuje się do systemu w początkowej fazie jego pracy. Model ten wykorzystany został do szacowania czasów odpowiedzi na żądania HTTP na podstawie obciążeń systemu lub jego elementów w pięciu spośród sześciu proponowanych systemów.

Część druga monografii (rozdz. 3) poświęcona została zagadnieniom podnoszenia jakości usług w klastrowych systemach webowych z kryterium czasowym. Zaproponowane zostały trzy systemy umożliwiające minimalizację czasów odpowiedzi na żądania HTTP.

Pierwszy, autorski system LFNRD przeznaczony jest do stosowania w klastrach serwerów webowych umieszczonych w jednej lokalizacji. W ramach opisu systemu przedstawiona została metoda dystrybucji żądań, zaprezentowany został projekt przełącznika webowego dystrybuującego żądania HTTP w klastrze serwerów WWW. Zaproponowany został algorytm dystrybucji, zgodnie z którym do obsługi żądania HTTP wybierany jest serwer oferujący najkrótsze czasy odpowiedzi na żądania.

Drugi opisany system GARDiB przeznaczony jest do stosowania w systemach globalnie rozproszonych klastrów serwerów webowych. W systemie tym wykorzystywane są serwery pośredniczące umieszczone na krawędzi Internetu, w pobliżu dużych skupisk potencjalnych użytkowników. Serwery pośredniczące przekierowują żądania użytkowników do serwisów lokalnych (klastrów serwerów webowych). W monografii opisana została metoda dystrybucji

żądań oraz projekt serwera pośredniczącego wykorzystującego algorytm dystrybucji żądań, który na podstawie informacji o stanie sieci rozległej oraz informacji o obciążeniach serwisów lokalnych wybierał ten serwis, dla którego czas przekazania odpowiedzi do serwera pośredniczącego jest najkrótszy.

Ostatnim prezentowanym autorskim systemem umożliwiającym podnoszenie jakości usług jest GARD. System ten w swej konstrukcji wykorzystuje serwisy lokalne, rozmieszczone w różnych lokalizacjach geograficznych oraz serwery dystrybucji żądań umieszczone w tych samych lokalizacjach co serwisy lokalne. Serwery dystrybucji żądań pośredniczą w wymianie żądań i obiektów między serwisami lokalnymi i klientami. Serwery te zmieniają zawartość dokumentów HTML w taki sposób, aby adresy obiektów zagnieżdżonych w stronach WWW oraz adresy hiperłączy wskazywały na te serwisy lokalne, które są w stanie dostarczyć klientowi wskazane obiekty w najkrótszym czasie. W opracowaniu przedstawiony został projekt serwera dystrybucji żądań GARD.

W trzeciej części monografii (rozdz. 4) opisane zostały trzy autorskie systemy webowe zapewniające jakość usług na wskazanym poziomie. Systemy te umożliwiają obsługę całych stron WWW w czasie nie dłuższym od zadanego. Pierwszym z zaprezentowanych systemów jest WEDF umożliwiający szeregowanie żądań HTTP na wejściu do serwisu webowego, w skład którego wchodzi jeden serwer webowy. W ramach opisu systemu WEDF zaprezentowana została metoda oraz przedstawiony został projekt serwera szeregującego wraz z odpowiednim algorytmem szacowania czasu odpowiedzi dla stron WWW.

Drugim zaprezentowanym autorskim systemem webowym jest system MLF, w którym wykorzystuje się lokalny klaster serwerów webowych. W ramach opisu systemu przedstawiona została metoda oraz projekt przełącznika webowego szeregującego i dystrybuującego żądania HTTP. W opisywanej metodzie żądania HTTP przydzielane są serwerom najbardziej obciążonym, które jednak są w stanie zrealizować obsługę żądań w wymaganym czasie.

Ostatnim prezentowanym autorskim systemem jest system GGARDiB, pracujący w środowisku rozproszonych geograficznie klastrów serwerów webowych. W systemie tym podobnie jak w systemie GARDiB wykorzystywane są serwery pośredniczące, umiejscowione w pobliżu dużych skupisk użytkowników. W ramach prezentacji projektu systemu opisana została metoda oraz przedstawiony został projekt serwera pośredniczącego i projekt współpracującego z serwerem pośredniczącym przełącznika webowego kontrolującego pracę klastra serwerów w serwisie lokalnym. W rozwiązaniu tym serwer pośredniczący przekierowuje żądania HTTP do tych serwisów lokalnych, które najszybciej obsłużą żądania, wskazując równocześnie przełącznikom webowym czasy, w których obsługa żądań musi zostać zakończona. Przełączniki webowe w odpowiedni sposób szeregują żądania oraz dystrybuują je pomiędzy serwery webowe.

Prezentowane w części drugiej oraz trzeciej projekty systemów stanowią komplet rozwiązań dla systemów webowych z kryterium czasowym. Wskazane zostały rozwiązania umożliwiające podnoszenie jakości usług oraz ich zapewnianie zarówno w małych systemach webowych, średnich jak i dużych.

Przedstawione w monografii rozwiązania i wyniki mogą mieć duże znaczenie praktyczne i poznawcze:

- stanowią podstawę systemów webowych z jakością usług uwzględniających kryterium czasowe,
- otwierają i stanowią podstawę nowej grupy systemów z zadanym czasem odpowiedzi (podgrupy systemów z kryterium czasowym),
- rozszerzają pulę algorytmów szeregowania i dystrybucji żądań; przedstawione algorytmy i metody mogą znaleźć oraz już znajdują swoje zastosowania nie tylko w problemach związanych z sieciami WWW, ale również w innych problemach spotykanych w informatyce i innych dziedzinach nauki,
- pokazują, jak metody i podejścia znane z ogólnej teorii systemów rozmytych i neuronowych można zastosować do rozwiązań problemów dystrybucji i szeregowania żądań w sieci WWW oraz jakie może to mieć korzyści w zakresie podnoszenia jak i zapewniania jakości usług,
- prezentowane w pracy projekty stanowić mogę podstawę metodyki projektowania systemów webowych z jakością usług.

W ramach zaprezentowanych rozwiązań wskazane zostały metody i algorytmy, które mogą być stosowane w kolejnych projektowanych systemach. Stąd też w naturalny sposób wypływają kierunki dalszych prac związanych z projektowaniem nie tylko systemów z kryterium czasowym, ale również systemów z kryterium biznesowym, w ramach których jakość oferowanych usług mogłaby zależeć od wymagań poszczególnych użytkowników tak, by zapewniona została jakość (czas odpowiedzi) dla każdego użytkownika odrębnie na precyzyjnie wskazanym poziomie.

Dodatkowo, wybrane rozwiązania mogą znaleźć zastosowania w projektowaniu systemów webowych wykorzystujących paradygmat SOA (ang. Service Oriented Architecture), w których przygotowanie treści prezentowanych użytkownikowi związane jest z realizacją licznych usług składowych. Pierwsze prace w tym kierunku wskazują, że podejście to jest obiecujące [80].

Ponadto, wydaje się również celowe zastosowanie omawianych rozwiązań w systemach dostarczania treści (systemach CDN), w których wykorzystuje się różne techniki buforowania treści. Wydaje się, że byłoby możliwe opracowanie systemu, który w skuteczny sposób zapewniałby czasy odpowiedzi dla klientów końcowych systemu.

LITERATURA

[1] ABDELZAHER T.F., SHINK K.G., BHATTI N.: Performance Guarantees for Web Server End-Systems: A Control-Theoretical Approach. IEEE Transactions on Parallel and Distributed Systems, vol. 13(1), Jan. 2002, pp. 80–96.

[2] AKAMAI, Produkty firmy Akamai. Dostęp 02.02.2011, dostępny w Internecie http://www.akamai.com.

[3] ALMEIDA J., DABUM., CAO P.: Providing differentiated levels of service in web content hosting. In Proc. of the First Workshop on Internet Server Performance, USA, 1998, pp. 91–102.

[4] ALSA'DEH A., YAHYA A. H.: Shortest Remaining Response Time Scheduling for Improved Web Server Performance. Lecture Notes in Business Information Processing, 2009, vol. 18, pp. 80–92.

[5] ANDERSSON M.: Introduction to Web Server Traffic Modeling and Control Research. Technical report, Sweden, Lund, publication code: Lutedx (Tets-7211)/1-27/(2005) 26, 2005.

[6] ANDREOLINI M., CASALICCHIO E., COLAJANNI M., MAMBELLI M.: A Cluster-Based Web System Providing Differentiated and Guaranteed Services. Cluster Computing, Jan. 2004, vol. 7, pp. 7–19.

[7] ANDREOLINI M., CASOLARI S., COLAJANNI M.: Autonomic request management algorithms for geographically distributed Internet-based systems. Proc. of 2nd IEEE International Conference on Self-Adaptive and Self-Organizing Systems (SASO 2008), Venice, Italy, Oct. 2008. IEEE Computer Society, Washington, DC, USA 2008, pp. 171–180.

[8] APACHE, strona projektu The Apache Software Foundation. Dostęp 02.04.2011, dostępny w Internecie http://www.apache.org/.

[9] APACHEBENCH, strona opisu oprogramowania ApacheBench. Dostęp 02.04.2012, dostępny w Internecie http://httpd.apache.org/docs/2.2/programs/ ab.html.

[10] APACHE JMETER, strona opisu oprogramowania Apache JMeter. Dostęp 02.04.2012, dostępny w Internecie http://jmeter.apache.org/.

[11] ARLITT M., JIN T.: Workload characterization of the 1998 World Cup Web site. Technical Report HPL-1999-35(R.1), Internet Systems and Applications Laboratory, HP Laboratories Palo Alto, USA, 1999.

[12] ARON M., SANDERS D., DRUSCHEL P., ZWAENPOEL W.: Scalable content-aware request distribution in cluster-based network servers. In Proc. of USENIX Annual Tech-

nical Conference ATEC 2000, San Diego, USA, USENIX Association, Berkeley, CA, USA, 2000, pp. 26–26.
[13] BARFORD P., CROVELLA M.: Generating Representative Web Workloads for Network and Server Performance Evaluation. In Proceedings of the 1998 ACM SIGMETRICS joint International Conference on Measurement and Modeling of Computer Systems, Madison, Wisconsin, USA, 1998, pp. 151–160.
[14] BARFORD P., BESTAVROS A., BRADLEY A., CROVELLA M.: Changes in Web access patterns characteristic and caching implications. World Wide Web Special Issue on Characterization and Performance Evaluation, 1999, vol. 2, pp. 15–28.
[15] BARFORD P., CROVELLA M.: Critical path analysis of TCP transactions. Proceedings of ACM SIGCOMM, Stockholm, September 2000. IEEE/ACM Transactions on Networking, 2000, pp. 127–138.
[16] BARROSO L.A., DEAN J., HOLZLE U.: Web search for a planet: The Google cluster architecture. IEEE Micro, vol. 23(2), USA, 2003, pp. 22–28.
[17] BARTOLINI N., BONGIOVANNI G., SILVESTRI S.: Self-* through self-learning: Overload control for distributed web systems. Computer Networks, 2009, vol. 53(5), pp. 727–743.
[18] BELLMAN R.E., ZADEH L.A.: Decision Making in a fuzzy environment. Management Science, 1970, vol. 17(4), pp. 141–164.
[19] BENDER M., CHAKRABARTI S., MUTHUKRISHNAN S.: Flow and stretch metrics for scheduling continuous Job streams. In Proceedings of the 9th Annual ACM-SIAM Symposium on Discrete Algorithms SODA '98, Society for Industrial and Applied Mathematics, Philadelphia, USA, 1998, pp. 270–279.
[20] BERNERS-LEE T., FIELDING R., IRVINE U., FRYSTYK H.: Hypertext Transfer Protocol – HTTP/1.0. RFC 1945, 1996, Dostęp 20.03.2010, dostępny w Internecie http://www.rfc-editor.org/rfc/rfc1945.txt.
[21] BERNERS-LEE T.: Information Management: A Proposal. Proposal for an information management system, CERN, 1989.
[22] 20 BŁAŻEWICZ J., CELLARY W., SŁOWIŃSKI R., WĘGLARZ J.: Badania operacyjne dla informatyków. Wydawnictwo Politechniki Poznańskiej, Poznań 1982, ISBN: 83-204-0519-X.
[23] BOONE B., HOECKE S.V., SEGHBROECK G.V., JONCHEERE N., JONCKERS V., TURCK F.D., DEVELDER C., DHOEDT B.: SALSA: QoS-aware load balancing for autonomous service brokering. In Proceedings of Journal of Systems and Software, New York, USA, 2010, vol. 83(3), pp. 446–456.
[24] BORZEMSKI L., ZATWARNICKI K.: Równoważenie obciążeń serwerów webowych – przegląd i badania wybranych rozwiązań systemowych. Oficyna Wydawnicza Politechniki Opolskiej, Zeszyty Naukowe Politechniki Opolskiej, seria Informatyka, Opole 2001, nr 282/2001, ss. 121–144.
[25] BORZEMSKI L., ZATWARNICKI K.: A Fuzzy Adaptive Request Distribution algorithm for cluster-based Web systems. Proceeding of 11th Euromicro Conference on Parallel Dis-

tributed and Network Based Processing, Genua, Włochy, 2003. IEEE Press, 2003, pp. 119–126.

[26] BORZEMSKI L., ZATWARNICKI K.: Using Adaptive Fuzzy-Neural Control to Minimize Response Time in Cluster-Based Web Systems. Lecture Notes in Computer Science, Springer-Verlag, Berlin Heidelberg, Germany, 2005, vol. 3528, pp. 63–68.

[27] BORZEMSKI L.: The use of data mining to predict Web performance. Cybernetics and Systems. 2006, vol. 37 (6), pp. 587–608.

[28] BORZEMSKI L., ZATWARNICKI K.: Performance Evaluation of Fuzzy-Neural HTTP Request Distribution for Web Clusters. Lecture Notes in Computer Science, Springer-Verlag, Berlin Heidelberg, Germany, 2006, vol. 4029, pp. 192–201.

[29] BORZEMSKI L., ZATWARNICKI K.: Fuzzy-Neural Web Switch Supporting Differentiated Service. Lecture Notes in Computer Science, Springer-Verlag, Berlin Heidelberg, Germany, 2006, vol. 4252, pp.195–203.

[30] BORZEMSKI L., CICHOCKI Ł., FRAS M., KLIBER M., NOWAK Z.: MWING: A Multiagent System for Web Site Measurements. Proceedings of the 1st KES International Symposium on Agent and Multi-Agent Systems: Technologies and Applications, KES-AMSTA, Springer-Verlag, Berlin Heidelberg, 2007, vol. 4496, pp. 278–287.

[31] BORZEMSKI L., ZATWARNICKA A., ZATWARNICKI K.: Środowisko symulacyjne dla globalnie rozproszonych systemów webowych. Sieci Komputerowe, tom 2, Aplikacje i zastosowania, praca zbiorowa pod red. A. Kwietnia, J. Obera, B. Pochopienia, P. Gaja. Wydawnictwo Komunikacji i Łączności, Warszawa 2007, ss. 23–32.

[32] BORZEMSKI L., ZATWARNICKA A., ZATWARNICKI K.: The Framework for Distributed Web Systems Simulation. Information Systems and Technology, Information Technology and Web Engineering: Models, Concepts and Challenges, Oficyna Wydawnicza Politechniki Wrocławskiej, Wrocław 2007, pp. 17–24.

[33] BORZEMSKI L., ZATWARNICKA A., ZATWARNICKI K.: Global Adaptive Request Distribution with Broker. Proceedings of 11th International Conference on Knowledge-Based and Intelligent Information & Engineering Systems, 12–14 September 2007, Vietri sul Mare, Italy. Lecture Notes in Artificial Intelligence, Springer-Verlag, Berlin Heidelberg, 2007, vol. 4693, pp. 271–278.

[34] BORZEMSKI L., ZATWARNICKI K., ZATWARNICKA A.: Adaptive and Intelligent Request Distribution for Content Delivery Networks. Cybernetics and Systems, 2007, vol. 38(8), pp. 837–857.

[35] BORZEMSKI L., NOWAK Z.: Using Autonomous System Topological Information in a Web Server Performance Prediction. Cybernetics and Systems, 2008, vol. 39 (7), pp. 753–769.

[36] BORZEMSKI L., ZATWARNICKA A., ZATWARNICKI K.: Local distributin algorithms of HTTP requests in content delivery systems. Information Systems Architecture and Technology: Web Information Systems, Models, Concepts & Challenges, Biblioteka Informatyki Szkół Wyższych, Wrocław 2008, pp. 13–24.

[37] BORZEMSKI L., ZATWARNICKI K., ZATWARNICKA A.: Badania nad metodą i algorytmami globalnej dystrybucji żądań HTTP w serwisach z serwerami pośredniczącymi. Współczesne aspekty sieci komputerowych, tom 1pod red. B. Pochopienia, P. Gaja, S. Kozielskiego, Wydawnictwo Komunikacji i Łączności, Warszawa 2008, ss. 169–178.

[38] BORZEMSKI L., ZATWARNICKA A., ZATWARNICKI K.: Taksonomia sieci dostarczania treści, Współczesne aspekty sieci komputerowych, tom 1 pod red. B. Pochopienia, P. Gaja, S. Kozielskiego, Wydawnictwo Komunikacji i Łączności, Warszawa 2008, ss. 179–188.

[39] BORZEMSKI L., ZATWARNICKA A., ZATWARNICKI K.: Method and algorithms of broker-based HTTP request global distribution. Theoretical and Applied Informatics, kwartalnik Komitetu Informatyki Polskiej Akademii Nauk, nr 1/2008, vol. 20(1), pp. 15–27.

[40] BORZEMSKI L., SUCHACKA G.: An approach to key customers and revenue oriented B2C service. Information Systems Architecture and Technology, IT Technologies in Knowledge Oriented Management Process, Oficyna Wydawnicza Politechniki Wrocławskiej, Wrocław 2009, pp. 251–262.

[41] BORZEMSKI L., KLIBER M., NOWAK Z.: Using Data Mining Algorithms in Web Performance Prediction. Cybernetics and Systems, 2009, vol. 40(2), pp. 176–187.

[42] BORZEMSKI L., SUCHACKA G.: Web Traffic Modeling for E-Commerce Web Server System. Communications in Computer and Information Science, Springer-Verlag, Berlin Heidelberg, 2009, vol. 39, pp. 151–159.

[43] BORZEMSKI L.: Web Mining Applications in E-commerce and E-services, Studies in Computational Intelligence, Springer-Verlag, Berlin Heidelberg, 2009, vol. 172, pp. 81–102.

[44] BORZEMSKI L., ZATWARNICKA A., ZATWARNICKI K.: Global distribution of HTTP requests using the fuzzy-neural decision-making mechanisms. Lecture Notes in Computer Science, Springer-Verlag, Berlin Heidelberg, 2009, vol. 5796, pp. 752–763.

[45] BOURKE T.: Wyrównywanie obciążeń serwerów. Wydawnictwo RM, Warszawa 2002, ISBN 83-7243-207-4.

[46] BRADEN R.: Requirements for Internet Hosts – Communication Layers. RFC 1122, Dostęp 21.05.1999, dostępny w Internecie ftp://ftp.rfc-editor.org/in-notes/rfc1122.txt.

[47] BUBNICKI Z.: Podstawy informatycznych systemów zarządzania. Wydawnictwo Politechniki Wrocławskiej, Wrocław 1993, ISBN 83-70-85059-6.

[48] BUBNICKI Z.: Teoria i algorytmy sterowania. Wydawnictwo Naukowe PWN, Warszawa 2002, ISBN 83-01-14414-9.

[49] BUYYA R., PATHAN M., VAKALI A.: Content Delivery Networks. Springer-Verlag, Berlin Heidelberg, Germany, 2008, ISBN 978-3-540-77886-8.

[50] CAO J., CLEVELAND W.S., YUAN GAO, JEFFAY K., SMITH F., WEIGLE M.: Stochastic models for generating synthetic HTTP source traffic. In Proceedings of Twenty-third Annual Joint Conference of the IEEE Computer and Communications Societies, INFOCOM 2004, 7–11 March 2004, Hong-Kong, pp. 1547–1558.

[51] CARDELLINI V., COLAJANNI M., YU P.S.: DNS dispatching algorithms with state estimators for scalable Web-server clusters. World Wide Web Journal, Baltzer Science, Aug. 1999, vol. 2(3), pp. 101–113.

[52] CARDELLINI V., CASALICCHIO E., COLAJANNI M., MAMBELLI M.: Web switch support for differentiated services. ACM Performance Evaluation Review, 2001, vol. 29(2), pp. 14–19.

[53] CARDELLINI V., CASALICCHIO E., COLAJANNI M., YU P.S.: The state of the art in locally distributed Web-server systems. ACM Computing Surveys, June 2002, vol. 34(2), pp. 263–311.

[54] CARDELLINI V., COLAJANNI M., YU P.S.: Request redirection algorithms for distributed web systems. IEEE Transactions on Parallel and Distributed Systems, April 2003, vol. 14(4), pp. 355–368.

[55] CARLSTROM J., ROM R.: Application-aware admission control and scheduling in Web servers. IEEE Proceedings of Twenty-First Annual Joint Conference on the IEEE Computer and Communications Societies, INFOCOM 2002, New York, 2002, vol. 2, pp. 506–515.

[56] CASALICCHIO E., COLAJANNI M.: A client-aware dispatching algorithm for web clusters providing multiple services. In Proceedings of the 10th International Conference on World Wide Web, WWW '01, Hong-Kong. ACM, New York, 2001, pp. 535–544.

[57] CHATTERJEE D., TARI Z., AND ZOMAYA A.Y.: A task-based adaptive TTL approach for Web server load balancing. In Proceedings of the 10th IEEE Symposium on Computers and Communications, IEEE Computer Society, Washington DC, USA, 2005. pp. 877–884.

[58] CHEN H., MOHAPATRA P.: Overload control in QoS-aware web servers. Computer Networks, Elsevier North-Holland, New York, USA, 2003, vol. 42(1), pp.119–133.

[59] CHENG A.: Real-Time Systems, Scheduling, Analysis and Verification. J.Wiley&Sons, 2002, ISBN 0-471-18406-3.

[60] CHERKASOVA L., KARLSSON M.: Scalable webserver cluster design with workload-aware request distribution strategy WARD. Proceedings of the Third International Workshop on Advanced Issues of E-Commerce and Web-Based Information Systems (WECWIS'01), San Jose, CA, USA. IEEE Computer Society, Washington, USA, 2001, pp. 212–221.

[61] CHERKASOVA L., AND PHAAL P.: Session-based admission control: A mechanism for peak load management of commercial Web sites. IEEE Transactions on Computers, 2002, vol. 51(6), pp. 669-685.

[62] CHOI E.: Performance test and analysis for an adaptive load balancing mechanism on distributed server cluster systems. Future Generation Computer Systems, vol. 20(2), Elsevier Science Publishers B.V., Amsterdam, The Netherlands, 2004, pp. 237–247.

[63] CIARDO G., RISKA A., SMIRNI E.: EQUILOAD: a load balancing policy for clustered web servers. Performance Evaluation, Elsevier Science Publishers, Amsterdam, The Netherlands, 2001, vol. 46(2–3), pp. 101–124.

[64] CISCO DISTRIBUTED DIRECTOR, opis produktu. Dostęp 21.01.2011, dostępny w Internecie http://www.cisco.co/en/US/products/h/contnetw/ps813/products_tech_note 09186a00801fa9dd.shtml.
[65] CISCO CSS 11500, opis produktu. Dostęp 21.01.2011, dostępny w Internecie http://www.cisco.com/en/US/products/hw/contnetw/ps792/.
[66] CISCO BOOMERANG, opis produktu. Dostęp 21.01.2011, dostępny w Internecie http://www.cisco.com/en/US/docs/ios/12_2t/12_2t8/feature//guide/ftdrpcli.htm.
[67] COLAJANNI M., YU P.S., CARDELLINI V.: Dynamic load balancing in geographically distributed heterogeneous Web-servers. Proceedings of IEEE 18th International Conference on Distributed Computing Systems (ICDCS'98), Amsterdam, The Netherlands, May 1998, pp. 295–302.
[68] COLAJANNI M., YU P.S., CARDELLINI V.: Scalable Web-Server systems: architectures, models and load balancing algorithms. Tutorial presented at ACM SIGMETRICS 2000, Santa Clara, USA, June 2000.
[69] COLAJANNI M., YU P.S.: A Performance Study of Robust Load Sharing Strategies for Distributed Heterogeneous Web Server Systems. IEEE Transactions on Knowledge and Data Engineering, NJ, USA, Mar. 2002, vol. 14(2), pp. 398–414.
[70] CSIM19, strona produktu CSIM19 firmy Mesquite Software. Dostęp 11.05.2011, dostępny w Internecie http://www.mesquite.com.
[71] CZACHÓRSKI T., FOURNEAU J.M., NOWAK S.: Symulacyjne badania routingu w sieciach optycznych. Wydawnictwo Politechniki Śląskiej, Studia Informatica, Gliwice 2002, vol. 23(2A), ss. 155–163
[72] CZACHÓRSKI T., PERKERGIN F.: A diffusion approximation model of web servers. In Proceedings of IFIP TC6/WG6.4 Workshop on Internet Technologies. Applications and Societal Impact (WITASI 2002), Wrocław, Poland, Kluwer Academic Publishers, Boston 2002, pp. 83–92.
[73] DRIANKOV D., HELLENDOORN H., REINFRANK M.: Wprowadzenie do sterowania rozmytego. WNT Warszawa, 1996, ISBN 83-204-2030-2.
[74] DUCH W., KORBICZ J., RUTKOWSKI L., TADEUSIEWICZ R.: Biocybernetyka i inżynieria biomedyczna. Tom 6, Sieci neuronowe. Akademicka Oficyna Wydawnicza EXIT, Warszawa 2000, ISBN 83-87674-18-4.
[75] F5, Big-IP, opis produktu. Dostęp 10.05.2011, dostępny w Internecie http://www.f5.com/products/big-ip/.
[76] F5, Global Load Balancing Solutions. Dostęp 10.05.2011, dostępny w Internecie http://www.f5.com/solutions/availability/global-load-balancing/.
[77] FAOUR A., MANSOUR N.: Weblins: A scalable www cluster-based server. Advances in Engineering Software, vol. 37, Elsevier Science Ltd., Oxford, UK, 2006, pp. 11–19.
[78] FIELDING R., IRVINE U., GETTYS J., MOGUL J., FRYSTYK H., BERNERS-LEE T.: Hypertext Transfer Protocol – HTTP/1.1, Rfc 2068. Dostęp 10.02.1999, dostępny w Internecie http://www.rfc-editor.org/.

[79] FIELDING R., GETTYS J., MOGUL J., FRYSTYK H., MASINTER L., BERNERS-LEE T.: Hypertext Transfer Protocol – HTTP/1.1, RFC 2616. Dostęp 10.12.1999, dostępny w Internecie http://www.rfc-editor.org/, 1999.

[80] FRAŚ M., ZATWARNICKA A., ZATWARNICKI K.: Fuzzy-Neural Controller in Service Requests Distribution Broker for SOA-Based Systems. Computer Networks, Communications in Computer and Information Science, Springer-Verlag, Berlin Heidelberg, 2010, vol. 79, pp. 121–130.

[81] GILLY K., JUIZ C., PUIGJANER R.: An up-to-date survey in web load balancing. Kluwer Academic Publishers, Hingham, World Wide Web, vol. 14(2), Ma, USA, 2011, pp. 105–131.

[82] GONT F., YOUTCHENKO A.: On the Implementation of the TCP Urgent Mechanism, RFC 6093. Dostęp 10.04.2011, dostępny w Internecie ftp://ftp.rfc-editor.org/in-notes/rfc6093.txt.

[83] GRZECH A., RYGIELSKI P., ŚWIĄTEK P.: Simulation environment for delivering quality of service in systems based on service-oriented architecture paradigm. Performance modelling and evaluation of heterogeneous networks, 6th Working International Conference HET – NETs 2010, Gliwice 2010. pp. 89–97.

[84] GUILLERMO N., MANIC M.: NFuSA – Neuro-Fuzzy Algorithm for Sparing in RAID Systems. The 33rd Annual Conference of the IEEE Industrial Electronics Society IECON07, Taipei, Taiwan, 2007, pp. 632–637.

[85] HAKATA T., MASUDA J.: Fuzzy control of cooling system utilizing heat storage. Proceedings of 1st International Conference on Fuzzy Logic and Neural Networks, Iizuka, Japan, 1990, pp. 77–80.

[86] HARCHOL-BALTER M., CROVELLA M., MURTA C.D.: On choosing a task assignment policy for a distributed server system. Journal of Parallel and Distributed Computing, vol. 59(2), Orlando, USA, 1999, pp. 204–228.

[87] HARCHOL-BALTER M., SCHROEDER B., BANSAL N., AGRAWAL M.: Size-based scheduling to improve web performance. ACM Transactions on Computer Systems, 2003, vol. 21 (2), pp. 207–233.

[88] HICKSON I.: HTML5, A vocabulary and associated APIs for HTML and XHTML. W3C Working Draft. Dostęp 25.05.2011, dostępny w Internecie http://www.w3.org/TR/html5/.

[89] HONG Y.S., NO J.H., KIM S.Y.: DNS-Based Load Balancing in Distributed Web-server Systems. Proceedings of the Fourth IEEE Workshop on Software Technologies for Future Embedded and Ubiquitous Systems, and the Second International Workshop on Collaborative Computing, Integration, and Assurance (SEUS-WCCIA'06), IEEE Computer Society, Washington, USA, 2006, pp. 251–254.

[90] HORIKAWA S., FURUHASHI T., UCHIKAWA Y.: On fuzzy modeling using fuzzy neural networks with the back-propagation algorithm. IEEE Transactions on Neural Networks, vol. 3(5), Los Alamitos, USA, 1992, pp. 801–806.

[91] HUANG C., ABDELZAHER T.: Towards content distribution networks with latency guarantees. In Proceedings of the 12th IEEE International Workshop on Quality of Service, IWQOS 2004, 7 June 2004, pp. 181–192.

[92] HUITEMA C., WEERAHANDI S.: Internet measurements: the rising tide and the DNS snag. In Proceedings of the ITC Specialist Seminar, IP Traffic Measurement, Modeling and Management, Monterey, CA, USA, 18–20 September, 2000, pp.130–145.

[93] IIS, strona projektu „Internet Information Services". Dostęp 15.04.2011, dostępny w Internecie http://www.iis.net/.

[94] IBM Network Dispatcher, opis produktu. Dostęp 15.04.2011, dostępny w Internecie http://www.redbooks.ibm.com /abstracts/sg246172.html.

[95] IBM WebSphere, opis produktu. Dostęp 15.04.2011, dostępny w Internecie http://www-306.ibm.com /software/info1/websphere, 2011.

[96] ISO/IEC 7498-1, Informational technology – Open Systems Interconnection – Basic Reference Model: The Basic Model, ISO, ref. num ISO/IEC 7498-1:1994(E), 1994.

[97] JANIAK A.: Wybrane problemy i algorytmy szeregowania zadań i rozdziału zasobów. Akademicka Oficyna Wydawnicza PLJ, Warszawa 1999, ISBN 83-7101-415-5.

[98] JÓZEFCZYK J.: Wybrane problemy podejmowania decyzji w kompleksach operacji. Wydawnictwa Politechniki Wrocławskiej, Wrocław 2001, ISBN 16408969-2001.

[99] KANDEL A.: Fuzzy Techniques in Pattern Recognition. J.Wiley&Sons, N.Y., USA, 1982, ISBN 0-4710-9136-7.

[100] KANODIA V., KNIGHTLY E.W.: Multi-class latency-bounded web services. In Proceedings of the 8th International Workshop on Quality of Service (IWQOS), 5–7 Jun 2000, Pittsburgh, PA, 2000, pp. 231–239.

[101] KARTALOPOULOS S.V.: Understanding Neural and Fuzzy Logic: Basic Concepts and Applications. Wiley-IEEE Press, N.Y., USA, 1995, ISBN 0-7803-1128-0.

[102] KAZIENKO P.: Associations: Discovery, Analysis and Applications. Oficyna Wydawnicza Politechniki Wrocławskiej, Wrocław 2008, ISBN 97-8837-49344-42.

[103] KRISHNAMURTHY B., REXFORD J.: HTTP/1.1, networking protocols, caching, and traffic measurement. Addison-Wesley, Boston, USA, 2001, ISBN 978-0-2017-1088-5.

[104] KOSKO B.: Neural Networks and Fuzzy Systems: A Dynamical Systems Approach to Machine Intelligence. Prentice Hall, USA, 1992, ISBN 0-13611-4350-103.

[105] KORBICZ J., OBUCHOWICZ A., UCIŃSKI D.: Sztuczne sieci neuronowe. Podstawy i zastosowania. Akademicka Oficyna Wydawnicza PLJ, Warszawa 1994, ISBN: 83-7101-197-0.

[106] KUNCHEVA L.I.: Fuzzy Classifier Design. Physica Verlag Heidelberg, 2000, ISBN 37-9081-298-6.

[107] LARSEN P.M.: Industrial application of fuzzy logic control. International Journal on Man-Machine Studies, 1980, vol. 12(1), pp. 3–10.

[108] LEE C.C.: Fuzzy Logic in Control Systems: Fuzzy Logic Controllers, IEEE Transactions on Systems, Man and Cybernetics, 1990, vol. 20(2), pp. 405–435.

[109] LINUX VIRTUAL SERVER, opis projektu. Dostęp 13.03.2011, dostępny w Internecie http://www.linuxvirtualserver.org/.

[110] LIU H.H., CHIANG M.L.: TCP rebuilding for content-aware request dispatching in web clusters. Journal of Internet Technology, 2005, vol. 6, pp. 231–240.

[111] LIU H.H., CHIANG M.L., WU M.C.: Efficient support for content-aware request distribution and persistent connection in Web clusters. Software: Practise and Experience, 2007, vol. 37(11), pp. 1215–1241.

[112] MAJUMDER D.: Fuzzy Mathematical Approach to Pattern Recognition. J.Wiley&Sons, New York, 1986, ISBN 0-470-27463-8.

[113] MALL R.: Real-Time Systems, Theory and Practice. Prentice Hall, 2009, ISBN 81-3170-069-0.

[114] MAMDANI E.H., ASSILIAN N.S.: A case study on the application of fuzzy set theory to automatic control. Proceedings of IFAC Stochastic Control Symposium, Budapest, 1974, pp. 33–45.

[115] MAMDANI E.H.: Application of fuzzy logic to approximate reasoning using linguistic synthesis. IEEE Transactions on Computers, vol. C-26(12), 1977, pp. 1182–1191.

[116] MAMDANI E.H.: Twenty years of fuzzy control: Experiences gained and lessons learnt. Second IEEE International Conference on Fuzzy Systems, San Francisco, USA, 1993, vol. 1, pp. 339 – 344.

[117] MARKS R.J.: Fuzzy Logic Technology and Applications. IEEE Technology Update Series, 1994.

[118] MARSZAŁEK G., ZATWARNICKA A., ZATWARNICKI K.: Aplikacja monitorująca sieć z wykorzystaniem algorytmu Cristiana. Techniczne i teoretyczne aspekty współczesnych sieci komputerowych, Wydawnictwo Komunikacji i Łączności, Warszawa 2009, ss. 69–78.

[119] MARSZAŁEK G., ZATWARNICKA A., BORZEMSKI L., ZATWARNICKI K.: Badania opóźnień pakietów w sieci z wykorzystaniem algorytmu Cristiana. Techniczne i teoretyczne aspekty współczesnych sieci komputerowych, Wydawnictwo Komunikacji i Łączności, Warszawa 2009, ss. 59–68.

[120] MARSZAŁEK G., ZATWARNICKA A., BORZEMSKI L., ZATWARNICKI K.: An Application for monitoring of hosts In local Or global Network based on Cristian's algorithm. Information Systems Architecture and Technology, Advances In Web-Age Information Systems, Oficyna Wydawnicza Politechniki Wrocławskiej, Wrocław 2009, pp. 47–57.

[121] MCCABE D.: Network Analysis, Architecture, and Design. 3rd Edition. Morgan Kaufmann, Boston, USA, 2007, ISBN 978-0-12-370480-2.

[122] MAZZOLLA M., MIRANDOLA R.: QoS Analysis For Web Service Applications: A Survey Of Performance-Oriented Approaches From An Architectural Viewpoint. Technical Report UBLCS-2010-05, University of Bologna, Italy, 2010.

[123] MEGAPANEL, Badanie Megapanel PBI/Gemius realizowane przez Polskie Badania Internetu Sp. z o.o. i Gemius S.A. Dostęp 10.02.2009, dostępny w Internecie http://www.panel.pbi.org.pl/wyniki.php.

[124] MEHTA H., KANUNGO P., CHANDWANI M.: Performance enhancement of scheduling algorithms in clusters and grids using improved dynamic load balancing techniques. In Proceedings of the 20th International Conference Companion on World Wide Web (WWW '11), Hyderabad, India. ACM, New York, USA, 2011, pp. 385–390.

[125] MENASCE D.A., FONSECA R., ALMEIDA V.A.F., MENDES M.A.: Resource management policies for e-commerce servers. SIGMETRICS Performance Evaluation Review, Mar. 2000, vol. 27(4), pp. 27–35.

[126] MENASCE D.A., ALMEIDA V.A.F.: Capacity planning for Web performance. Metrics, models, and methods. Prentice Hall PTR, New Jersey, 2002, ISBN 0-1306-5903-7.

[127] MOON J., KIM M.: Dynamic load balancing method based on DNS for distributed web systems. In E-Commerce and Web Technologies, Lecture Notes in Computer Science, Springer-Verlag, Berlin Heidelberg, 2005, vol. 3590, pp. 238–247.

[128] MURTA C.D., CORLASSOLI T.P.: Fastest connection first: A new scheduling policy for web servers. In Proceedings of the 18th International Teletraffic Congress (ITC-18), Brazil, September 2003, pp. 122–130.

[129] NAKAI A.M., MADEIRA E., BUZATO L.E.: DNS-based load balancing for web services. In Proceedings of the 6th International Conference on Web Information Systems and Technologies, Spain, 2010, pp. 95–100.

[130] NAKAI A.M., BUZATO L.E.: Improving the QoS of Web Services via Client-Based Load Distribution. In Proceedings of the XXIX Simpósio Brasileiro de Redes de Computadores e Sistemas Distribuídos, Brasil, 2011, pp. 617–629.

[131] OK M., PARK M.S.: Distributing requests by (around k)-bounded load-balancing in web server cluster with high scalability. IEICE-Transactions on Information and Systems, 2006, vol. E89-D(2), pp. 663–672.

[132] OSOWSKI S., LINH T.H.: Selforganizing neural and neurofuzzy networks – a comparative study. Archives of Electrical Engineering, Polish Academy of Science, 1999, vol. 48(1–2), pp. 199–218.

[133] PAI V.S., ARON M., BANGA G., SVENDSEN M., DRUSCHEL P., ZWAENPOEL W., NAHUM E.: Locality-aware request distribution in cluster-based network servers. In Proceedings of 8th ACM Conference on Architectural Support for Programming Languages and Operating Systems, San Jose, October 1998. SIGOPS Operating System Review, ACM, New York, USA, 1998, vol. 32(5), pp. 205–216.

[134] PAN J., HOU Y.T., LI B.: An overview of DNS-based server selections in content distribution networks. Computer Networks, Elsevier North-Holland, New York, USA, 2003, vol. 43(6), pp. 695–711.

[135] PARK S. Y., PARK D., LEE J., CHO J.W.: Efficient Inter-Backend prefetch algorithms in cluster-based web servers. In Proceedings of the International Conference/Exhibition on High Performance Computing (HPC ASIA), Australia, 2001, pp. 104–122.

[136] PATHAN M., BUYYA R., VAKALI A.: CDNs: State of the Art, Insights, and Imperatives. Content Delivery Networks, Lecture Notes in Electrical Engineering, Springer-Verlag, Berlin Heidelberg Germany, 2008, vol. 9(1), pp. 3–32.

[137] PATHAN M., BUYYA R.: Resource discovery and request-redirection for dynamic load sharing in multi-provider peering content delivery networks. Journal of Network and Computer Applications, Academic Press Ltd., London, UK, 2009, vol. 32(5), pp. 976–990.

[138] PEDRYCZ W.: Fuzzy Control and Fuzzy Systems. J.Willey&Sons, Toronto, 1993, ISBN 0-8638-0131-5.

[139] PETRUCCI V., LOQUES O.: A Framework-based Approach to Support Dynamic Adaptation of Web Server Clusters. 10th Brazillian Workshop on Real-Time and Embedded Systems (WTR), Rio de Janeiro-RJ, Brazil, 2008, pp. 67–76.

[140] PIEGAT A.: Modelowanie i sterowanie rozmyte. Akademicka Oficyna Wydawnicza EXIT, Warszawa 1999, ISBN 83-87674-14-1.

[141] PILIŃSKI M.: Universal Network Trainer. Proceedings of the Second Conference Neural Networks and Their Applications, Częstochowa, 1996, vol. 2, pp. 383–391.

[142] POPPER K.R.: Logika odkrycia naukowego. Przekład Nikłaś U., Aletheia, 2002, ISBN 83-89372-01-0.

[143] POSTEL J.: Transmission Control Protocol – TCP. RFC 793, 1996. Dostęp 10.03.2009, dostępny w Internecie http://www.rfc-editor.org/rfc/rfc793.txt.

[144] RADWARE, Enhancing Web User Experience with Global Server Load Balancing, White Paper, materiały informacyjne. Dostęp 11.01.2009, dostępny w Internecie http://www.radware.com/Resources/rclp.aspx.

[145] RADWARE, Radware Appdirector, opis produktu. Dostęp 11.01.2011, dostępny w Internecie http://www.radware.com/Solutions/Enterprise/DataCenter/BusinessContinuity.aspx.

[146] RANJAN S., KNIGHTLY E.: High-performance resource allocation and request redirection algorithms for web clusters. IEEE Transactions on Parallel Distributed Systems, IEEE Press, Piscataway, USA, 2008, vol. 19(9), pp. 1186–1200.

[147] RAMAKRISHNAN K., FLOYD S., BLACK D.: The Addition of Explicit Congestion Notification (ECN) to IP. RFC 3168, Dostęp 10.01.2011, dostępny w Internecie ftp://ftp.rfc-editor.org/in-notes/rfc3168.txt.

[148] RISKA A., SUN W., SMIRNI E., CIARDO G.: AdaptLoad: effective balancing in clustered web servers under transient load conditions. In Proceedings of the 22nd International Conference on Distributed Computing Systems (ICDCS 2002), Vienna, Austria, July, 2002, pp. 103–111.

[149] ROPERO J., LEÓN C., CARRASCO A., GÓMEZ A., RIVERA O.: Fuzzy Logic Applications for Knowledge Discovery: a Survey. International Journal of Advancements in Computing Technology, vol. 3(6), 2011, pp. 187–198.

[150] RUTKOWSKA D.: Inteligentne systemy obliczeniowe. Algorytmy genetyczne i sieci neuronowe w systemach rozmytych. Akademicka Oficyna Wydawnicza PLJ, Warszawa 1997, ISBN 83-7101-361-2.

[151] RUTKOWSKA D., PILIŃSKI M., RUTKOWSKI L.: Sieci neuronowe, algorytmy genetyczne i systemy rozmyte. Wydawnictwa Naukowe PWN, Warszawa–Łódź, 1997, ISBN 83-0112-304-4.

[152] SHARIFIAN S., AKBARI M.K., MOTAMEDI S.A.: An Intelligence Layer-7 Switch for Web Server Clusters. 3rd International Conference: Sciences of Electronic, Technologies of Information and Telecommunications. Tunisia, 2005 March, pp. 27–31.

[153] SATAKE S., INAI H.: Special issue on internet architecture technology papers: A nonprobabilistic server selection method based on periodically obtained load information for web server clusters. Electronics and Communications in Japan, Part I: Communications, 2006, vol. 89(10), pp. 1–12.

[154] SCHROEDER B., HARCHOL-BALTER M.: Web servers under overload: how scheduling can help. ACM Transactions on Internet Technology (TOIT), New York, USA, 2006, vol. 6(1), pp. 20–52.

[155] SHAN Z., LIN C., MARINESCU D.C., YANG Y.: Modeling and performance analysis of QoS-aware load balancing of web-server clusters. Computer Networks, Elsevier North-Holland, New York, USA, 2002, vol. 40(2), pp. 235–256.

[156] SHARIFIAN S., MOTAMEDI S., AKBARI M.: A content-based load balancing algorithm with admission control for cluster web servers. Journal of Future Generation Computer Systems, Elsevier Science Publishers B.V., Amsterdam, The Netherlands, October 2008, vol. 24(8), pp. 775–787.

[157] SHARMA S., SINGH S., SHARMA M.: Performance Analysis of Load Balancing Algorithms. World Academy of Science, Engineering and Technology Journal, 2008, vol. 38, pp. 269–272.

[158] SHIAN-TANG T.: Design of Self-Learning Fuzzy System by GA Approach. In Proceedings of the Eighth International Conference on Intelligent Systems Design and Applications (ISDA'08), IEEE Press, Taiwan, November 2008, vol. 2, pp. 313–318.

[159] SHIVARATRI N.G., KRUEGER P., SINGHAL M.: Load distributing for locally distributed systems. IEEE Computer Society Press, Los Alamitos, CA, USA, December 1992, vol. 25(12), pp. 33 – 44.

[160] SINGHMAR N., MATHUR V., APTE V., MANJUNATH D.: A Combined LIFO-Priority Scheme for Overload Control of E-commerce Web Servers. International Infrastructure Survivability Workshop, Lisbon, Portugal, December 2004, pp. 102–120.

[161] SIT Y.F., WANG C.L., LAU F.: Cyclone: A high-performance clusterbased web server with socket cloning. Cluster Computing, 2004, vol. 7(1), pp. 21–37.

[162] SU A.J., CHOFFNES D.R., KUZMANOVIC A., BUSTAMANTE F.E.: Drafting behind Akamai (travelocity-based detouring). Proceedings of the Conference on Applications, Technologies, Architectures and Protocols for Computer Communications (SIGCOMM'06), Pisa, Italy. ACM SIGCOMM computer Communication Review, New York, 2006, vol. 36(4), pp. 435–446.

[163] SUGENO M.: An Introductory Survey o Fuzzy Control. Information Sciences, July 1985, vol. 36(1–2), pp. 59–83.

[164] ŚWIĄTEK J.: Wybrane zagadnienia identyfikacji statycznych systemów złożonych. Oficyna Wydawnicza Politechniki Wrocławskiej, Wrocław 2009, ISBN 83-7493-4425.
[165] TADEUSIEWICZ R.: Sieci neuronowe. Akademicka Oficyna Wydawnicza, Warszawa 1993, ISBN 83-85769-03-X.
[166] TANENBAUM A.: Sieci komputerowe. Wydanie IV, tłumaczenie: Andrzej Grażyński, Adam Jarczyk, Helion, Gliwice 2004, ISBN 8373615571.
[167] TANG X., CHANSON S.T.: On caching effectiveness of web clusters under persistent connections. Journal of Parallel and Distributed Computing. Scalable web services and architecture, Academic Press, Orlando, USA, October 2003, vol. 63(10), pp. 981–995.
[168] ŢARIOV A., MĄKA T.: Wprowadzenie do modelowania sygnałów telekomunikacyjnych w środowisku Matlab-Simulink, Wydawnictwo Uczelniane Politechniki Szczecińskiej, Szczecin, Polska, 2008, ss.1–136.
[169] VAKALI, A., PALLIS G.: Content delivery networks: status and trends. IEEE Internet Computing, IEEE Educational Activities Department, Piscataway, USA, November 2003, vol. 7(6), pp. 68–74.
[170] VENKETESH P., VENKATESAN R.: A Survey on Applications of Neural Networks and Evolutionary Techniques in Web Caching. IETE Tech Rev, 2009, vol 26 (3), pp. 171–80.
[171] W3C, Publikacje World Wide Web Consortium, Dostęp 05.01.2011, dostępny w Internecie http://www.w3.org/.
[172] WEI J., ZHOU X., XU CH.Z.: Robust processing rate allocation for proportional slowdown differentiation on Internet servers. IEEE Transactions on Computers, IEEE Computer Society, Washington, USA, August 2005, vol. 54(8), pp. 964–977.
[173] WEI J., XU CH. Z.: eQoS: Provisioning of client-perceived end-to-end QoS guarantees in Web servers. IEEE Transactions on Computers, IEEE Computer Society, Washington, USA, December 2006, vol. 55(12), pp. 1543–1556.
[174] WERBOS P.: Beyond regression: new tools for prediction and analysis in the behavioral sciences. Ph.D. dissertation, Committee on Applied Mathematics, Harvard University, Cambridge, MA, USA, 1974.
[175] WGET, Opis oprogramowania WGet. Dostęp 05.02.2007, dostępny w Internecie http://www.gnu.org/software/wget/.
[176] WIDROW B., HOFF M.E.: Adaptive switching circuits. New York IRE WESCON Convention Record, 1960, pp. 96–104.
[177] WILLIAMS C.W.A., ARLITT M., BARKER K.: Web workload characterization: Ten years later. Web Content Delivery. Springer Springer-Verlag, Berlin Heidelberg 2005, pp. 3–21.
[178] YAGER R.R., ZADEH L.A.: Fuzzy sets, neural networks, and soft somputing. Thomson Learning, New York, USA, 1994, ISBN 04-4201-621-2.
[179] YAGER R.R., FILEV D.P.: Podstawy modelowania i sterowania rozmytego, tłumaczenie: Mazur C., Wańczuk T., Jankowski S., Wydawnictwa Naukowo-Techniczne, Warszawa 1995, ISBN 83-2041-909-3.

[180] YOKOTA H., KIMURA S., EBIHARA Y.: A proposal of DNS-based adaptive load balancing method for mirror server systems and its implementation. Proceedings of the 18th International conference on Advanced Information Networking and Applications (AINA'04), IEEE computer Society, Washington, USA, 2004, vol. 2, pp. 208–213.

[181] YUNFENG L., QINGSHENG Z., YUKUN C.: A request dispatching policy for Web server cluster. The 2005 IEEE International Conference on e-Technology, e-Commerce and e-Service, EEE '05, China, 2005, pp. 391 – 394.

[182] ZADEH L.A.: Fuzzy sets. Information and Control, 1965, vol. 8, pp. 338–353.

[183] ZADEH L.A.: Fuzzy algorithms. Information and Control, 1968, vol. 12, pp. 94–102.

[184] ZADEH L.A.: A rationale for fuzzy control. Journal of Dynamic System Measuring and Control, 1972, vol. 94, series G, pp. 3–4.

[185] ZADEH L.A.: Outline of a new approach to the analysis of complex systems and decision processes. IEEE Transactions on Syst., Man., Cyber., 1973, vol. SMC-3, No. 1, pp. 28–44.

[186] ZADEH L.A.: Fuzzy logic-computing with words. IEEE Transactions on Fuzzy Systems, IEEE Computational Intelligence Society, 1996, vol. 4(2), pp. 103–111.

[187] ZATWARNICKA A., ZATWARNICKI K.: Budowa symulatora serwisu webowego z wykorzystaniem pakietu CSIM. Oficyna Wydawnicza Politechniki Opolskiej, Zeszyty Naukowe Politechniki Opolskiej, seria Informatyka, Opole 2005, nr 302/2005 z. 2, ss. 107–125.

[188] ZATWARNICKA A., ZATWARNICKI K.: Adaptive HTTP request distribution in time-varying environment of globally distributed cluster-based Web system. Lecture Notes in Computer Science, Springer-Verlag, Berlin Heidelberg, 2011, vol. 6881, pp. 141–150.

[189] ZATWARNICKI K.: Algorytmy równoważenia obciążeń lokalnie rozmieszczonych serwerów WWW. Studia Informatica, Zeszyty Naukowe Politechniki Śląskiej, seria Informatyka, Wydawnictwo Politechniki Śląskiej, Gliwice 2001, vol. 22(1), ss. 147–162.

[190] ZATWARNICKI K.: Load balancing algorithms in locally distributed web systems. Proceedings of the 23rd conference Information Systems Architecture and Technology (ISAT 2001), International Scientific School Digital Economy Concepts, Tools and Applications, Oficyna Wydawnicza Politechniki Wrocławskiej, Wrocław 2001, pp. 227–234.

[191] ZATWARNICKI K.: Adaptacyjny algorytm równoważenia obciążeń lokalnie rozmieszczonych serwerów WWW. Studia Informatica, Zeszyty Naukowe Politechniki Śląskiej, seria Informatyka, Wydawnictwo Politechniki Śląskiej, Gliwice 2002, vol. 23(2B), ss. 59–66.

[192] ZATWARNICKI K.: Adaptacyjny algorytm dystrybucji żądań w klastrze serwerów WWW. Materiały V Konferencji Naukowej Inżynieria wiedzy i systemy ekspertowe, tom 2, 11–13 czerwca 2003 Wrocław, Oficyna Wydawnicza Politechniki Wrocławskiej, Wrocław 2003, ss. 310–318.

[193] ZATWARNICKI K., ZATWARNICKA A.: Modele oceny wydajności serwisów internetowych. Współczesne problemy sieci komputerowych, pod red. S. Węgrzyna, P. Pochopienia, T. Czachórskiego, Wydawnictwa Naukowo-Techniczne, Warszawa 2004, ss. 251–259.

[194] ZATWARNICKI K., ZATWARNICKA A.: Modelowanie wydajności serwisów internetowych. Materiały IV Krajowej Konferencji Multimedialne i Sieciowe Systemy Informacyjne (MiSSI 2004), Oficyna Wydawnicza Politechniki Wrocławskiej, Wrocław 2004, vol.1, ss. 189–198.

[195] ZATWARNICKI K.: Proposal of a neuro-fuzzy model of a WWW server. Proceedings of the 5th International Conference on Intelligent Systems Design and Applications (ISDA'05), IEEE Computer Society, Poland, Wrocław, 2005, pp. 141–146.

[196] ZATWARNICKI K.: Modele symulacyjne serwerów webowych – modele rozmyto-neuronowe. Wysoko wydajne sieci komputerowe. Zastosowanie i bezpieczeństwo, pod red. A. Kwietnia, A. Grzywaka, Wydawnictwo Komunikacji i Łączności, Warszawa, 2005, ss. 145–156.

[197] ZATWARNICKI K.: Propozycja rozmyto-neuronowego modelu serwera WWW. Zeszyty Naukowe Politechniki Opolskiej, seria Informatyka, Oficyna Wydawnicza Politechniki Opolskiej, Opole 2005, nr 302/2005 z. 2, ss. 205–217.

[198] ZATWARNICKI K., SOLGA P.: Badania charakterystyk polskich stron internetowych. Nowe technologie sieci komputerowych, pod red. B. Pochopienia, A. Kwietnia, A. Grzywaka, J. Klamki, tom 2, Wydawnictwa Komunikacji i Łączności, Warszawa 2006, ss. 211–218.

[199] ZATWARNICKI K., BORZEMSKI L.: Rozmyto-neuronowa dystrybucja żądań w klastrowych systemach webowych. Nowe technologie sieci komputerowych, pod red. S. Węgrzyna, L. Znamirowskiego, T. Czachórskiego, S. Kozielskiego, tom 1, Wydawnictwa Komunikacji i Łączności, Warszawa 2006, ss. 81–90.

[200] ZATWARNICKI K.: Metoda adaptacyjnej dystrybucji treści w sieciach CDN. Sieci Komputerowe, tom 2 Aplikacje i zastosowania, pod red. A. Kwietnia, J. Obera, B. Pochopienia, P. Gaja, Wydawnictwa Komunikacji i Łączności, Warszawa 2007, ss. 137–145.

[201] ZATWARNICKI K.: Web object characteristics. Information Systems and Technology, Information Technology and Web Engineering: Models, Concepts and Challenges, Oficyna Wydawnicza Politechniki Wrocławskiej, Wrocław 2007, pp. 25–32.

[202] ZATWARNICKI K., ZATWARNICKA A., BORZEMSKI L.: Metoda szacowania czasu odpowiedzi na żądania HTTP po stronie serwisu webowego. Sieci komputerowe, tom 2 Aplikacje i zastosowania, pod red. A. Kwietnia, J. Obera, B. Pochopienia, P. Gaja, Wydawnictwa Komunikacji i Łączności, Warszawa 2007, ss. 127–136.

[203] ZATWARNICKI K., SOLGA P.: Wyznaczanie charakterystyk stron światowego Internetu jako parametrów generatora żądań HTTP. Sieci komputerowe, tom 2 Aplikacje i zastosowania, pod red. A. Kwietnia, J. Obera, B. Pochopienia, P. Gaja, Wydawnictwa Komunikacji i Łączności, Warszawa 2007, ss. 117–126.

[204] ZATWARNICKI K.: Determination of parameters of parameters of web server simulation model. Information Systems Architecture and Technology, Web Information Systems, Models, Concepts & Challenges, Biblioteka Informatyki Szkół Wyższych, Oficyna Wydawnicza Politechniki Wrocławskiej, Wrocław 2008, pp. 25–36.

[205] ZATWARNICKI K.: Badania nad algorytmami globalnej dystrybucji żądań HTTP w sieci Internet, Współczesne aspekty sieci komputerowych, tom 1, pod red. B. Pochopienia, P. Gaja, S. Kozielskiego, Wydawnictwa Komunikacji i Łączności, Warszawa 2008, ss. 159–168.

[206] ZATWARNICKI K., BORZEMSKI L.: CDNs with Global Adaptive Request Distribution. Proceedings of the 12th international conference on Knowledge-Based Intelligent Information and Engineering Systems, Part II, Lecture Notes in Computer Science, Springer-Verlag, Berlin Heidelberg, 2008, vol. 5178, pp.117–124.

[207] ZATWARNICKI K.: Obsługa w sposób deterministyczny stron WWW przez serwis webowy. Techniczne i teoretyczne aspekty współczesnych sieci komputerowych, Wydawnictwa Komunikacji i Łączności, Warszawa 2009, ss. 59–68.

[208] ZATWARNICKI K.: Application of fuzzy-neural models in the distribution of HTTP requests in the local web server cluster. Information Systems Architecture and Technology, Advances In Web-Age Information Systems, Oficyna Wydawnicza Politechniki Wrocławskiej, Wrocław 2009, pp. 47–57.

[209] ZATWARNICKI K.: Deterministic processing of WWW pages by the Web service. Lecture Notes in Computer Science, Springer-Verlag, Berlin Heidelberg, 2009, vol. 5796, pp. 764–775.

[210] ZATWARNICKI K.: A cluster-based web system providing guaranteed service. Systems Science, Wrocław, 2009, vol. 35(4), pp. 69–80.

[211] ZATWARNICKI K.: Taksonomia systemów webowych z jakością usług. Wydawnictwa Komunikacji i Łączności, Warszawa 2010, ss. 109–119.

[212] ZATWARNICKI K.: Providing Web service of established quality with the use of HTTP requests scheduling methods. Agent and Multi-agent systems: Technologies and Applications, Lecture Notes in Artificial Intelligence, Springer-Verlag, Berlin Heidelberg 2010, vol. 6070, pp. 142–151.

[213] ZATWARNICKI K.: Neuro-Fuzzy models in global HTTP request distribution. Computational Collective Intelligence, Technologies and Applications, Lecture Notes in Computer Science, Springer-Verlag, Berlin Heidelberg 2010, vol. 6421, pp. 1–10.

[214] ZATWARNICKI K., ZATWARNICKA A., GASZ R., GRZESIK Ł.: Slow Start algorithm influence on effective throughput in the Internet. Information Systems Architecture and Technology, Networks and Networks' Services. Oficyna Wydawnicza Politechniki Wrocławskiej, Wrocław 2010, pp. 161–172.

[215] ZATWARNICKI K., WAŃKOWICZ A., ZATWARNICKA A.: Badanie wydajności klastrów serwerów WWW pracujących pod kontrolą programowych przełączników webowych. Wydawnictwa Komunikacji i Łączności, Warszawa 2010, ss. 99–108.

[216] ZATWARNICKI K., ZATWARNICKA A.: Method of parameters selection for fuzzy-neural system provided global HTTP distribution. Information Systems Architecture and Technology, New Development in Web-Age Information Systems. Oficyna Wydawnicza Politechniki Wrocławskiej, Wrocław 2010, pp. 65–76.

[217] ZATWARNICKI, K.: Identification of Web server. Communications in Computer and Information Science, Springer-Verlag, Berlin Heidelberg 2011, vol. 160, pp. 45–54.

[218] ZATWARNICKI K.: Adaptive request distribution in cluster-based Web system, Lecture Notes in Computer Science, Springer-Verlag, Berlin Heidelberg Springer, Berlin Heidelberg 2011, vol. 6881, pp. 42–51.

[219] ZATWARNICKI K.: Guaranteeing quality of service in globally distributed Web system with brokers, Lecture Notes in Artificial Intelligence, Springer-Verlag, Berlin Heidelberg 2011, vol. 6923, pp. 374–384.

[220] ZATWARNICKI K.: Dystrybucja żądań HTTP w klastrowych systemach webowych z wykorzystaniem sieci rozmyto-neuronowych. Pomiary Automatyka Kontrola, 2011, vol. 03/2011, ss. 305–307.

[221] ZATWARNICKI K., ZATWARNICKA A.: Pomiary wydajności popularnych serwerów webowych, Studia Informatica, Studia Informatica, Zeszyty Naukowe Politechniki Śląskiej, seria Informatyka, Wydawnictwo Politechniki Śląskiej, Gliwice 2011, vol. 32(3A), ss. 103–112.

[222] ZENG L., BENATALLAH B., NGU A., DUMAS M., KALAGNANAM J., CHANG H.: QoS-Aware Middleware for Web Services Composition. IEEE Transactions on software Engineering, IEEE Press Piscataway, USA, 2004, vol. 30(5), pp. 311–327.

[223] ZGRZYWA A.: Ocena wydajności systemów informacyjnych metodami kolejkowymi. Oficyna Wydawnicza Politechniki Wrocławskiej, Wrocław 1998, ISBN 83-7085-352-8.

[224] ZHANG X., BARRIENTOS M., CHEN J., SELTZER M.: HACC: An architecture for cluster-based web servers. In Proceedings of the 3rd USENIX Windows NT Symposium, vol. 3, USENIX Association Berkeley, USA, 1999, pp. 155–164.

[225] ZHANG Q., RISKA A., SUN W., SMIRNI E., CIARDO G.: Workloadaware load balancing for clustered web servers. IEEE Transactions on Parallel and Distributed Systems, IEEE Press, 2005, vol. 16(3), pp. 219–233.

[226] ZHANG Q., MI N., RISKA A., SMIRNI E.: Load unbalancing to improve performance under autocorrelated traffic. In Proceedings of the 26th IEEE International Conference on Distributed Computing Systems, 4–7 July, Lisboa, Portugal. IEEE Computer Society, USA, 2006, pp. 20–32.

[227] ZHONG F.,YIN S.: Response Delay Predictive and Load Scheduling Control for the Cluster Server. Computer Systems & Applications, China, 2007, vol. 7, pp. 26–29.

[228] ZHOU X., WEI J., XU C.Z.: Resource Allocation for Session-Based Two-Dimensional Service Differentiation on e-Commerce Servers. IEEE Transactions on Parallel and Distributed Systems, IEEE Press, Piscataway, USA, August 2006, vol. 17(8), pp. 838–850.

SYSTEMY WEBOWE Z JAKOŚCIĄ USŁUG UWZGLĘDNIAJĄCE KRYTERIUM CZASOWE
Problemy projektowania

Streszczenie

W monografii przedstawione zostały zagadnienia projektowania systemów webowych z jakością usług ze szczególnym uwzględnieniem systemów, w których bierze się pod uwagę czas realizacji zadań jako kryterium jakości.

Celem pracy było opracowanie grupy systemów webowych z kryterium czasowym. Proponowane systemy powinny umożliwiać podnoszenie jakości oferowanych usług poprzez minimalizację czasów odpowiedzi na żądania HTTP oraz utrzymywanie czasów odpowiedzi nie dłuższych niż założone dla całych stron WWW dla małych serwisów webowych zawierających jeden lub kilka serwerów WWW oraz dużych serwisów zawierających do kilkudziesięciu serwerów WWW.

W rozdziale pierwszym monografii opisane zostały zagadnienia dotyczące budowy, sposobu działania oraz taksonomii systemów webowych z jakością usług. Wskazana została potrzeba rozwijania systemów z kryterium czasowym minimalizujących czas odpowiedzi oraz systemów z zadanym czasem odpowiedzi.

W rozdziale drugim przedstawione zostało opracowanie rozmyto-neuronowego modelu systemu obsługi żądań HTTP umożliwiającego szacowanie czasów odpowiedzi na żądania HTTP. Model ten zastosowany został w pięciu spośród sześciu prezentowanych dalej systemów webowych.

W rozdziale trzecim zaprezentowane zostały projekty trzech systemów webowych umożliwiających minimalizowanie czasu odpowiedzi na pojedyncze żądania HTTP:

- system LFNRD przeznaczony do stosowania w klastrach serwerów webowych umieszczonych w jednej lokalizacji,
- system GARDiB przeznaczony do stosowania w systemach globalnie rozproszonych klastrów serwerów webowych. W systemie tym wykorzystywane są serwery pośredniczące umieszczone na krawędzi Internetu, w pobliżu dużych skupisk potencjalnych użytkowników,
- system GARD przeznaczony do stosowania w systemach globalnie rozproszonych klastrów serwerów webowych. W systemie tym nie są wykorzystywane serwery pośredniczące.

W rozdziale czwartym przedstawione zostały projekty trzech systemów webowych z zadanym poziomem jakości usług:

- system WEDF umożliwiający szeregowanie żądań HTTP na wejściu do serwisu webowego, w skład którego wchodzi jeden serwer webowy,
- system MLF, w którym wykorzystuje się lokalny klaster serwerów webowych,
- system GGARDiB, pracujący w środowisku rozproszonych geograficznie klastrów serwerów webowych.

Jakość działania systemów prezentowanych w rozdziałach 3 i 4 oceniona została po przeprowadzeniu badań symulacyjnych.

Monografia podsumowana została w ostatnim piątym rozdziale. Przedstawione zostały tu również plany opracowania kolejnych systemów typu e-comerce gwarantujących zróżnicowaną jakość pracy oraz systemów typu SOA oraz CDN.

QUALITY-BASED TIME-AWARE WEB SYSTEMS
Design Problems

Summary

The monograph has presented some solutions to problems in designing quality-based Web systems focused on systems taking into account response times as quality criteria.

The purpose assumed in the work concerned development of quality-based time-aware Web systems. The proposed systems should improve quality of Web service by minimizing HTTP request response time and keeping the page response time within the established boundaries both for small Web systems containing few Web servers as well as complex Web systems conteining globally distributed Web clusters.

In Chapter 1, the construction, operation and taxonomy of quality-based Web systems has been presented. The need to develop time-aware Web systems minimizing request response time and operating on desired level of quality has been substantiated.

Chapter 2 has offered the desing of a neuro-fuzzy model of HTTP request service anabling estimation of HTTP request response times. The model has been used in five out of six presented Web systems.

In Chapter 3, the design of three Web systems enabling minimalization of HTTP request response time has been presented:

- the LFNRD system designed to control a cluster of Web servers,
- the GARDiB system designed to control globally distributed Web clusters with the use of broker servers,
- the GARD system controlling globally distributed Web clusters withought any broker server.

Chapter 4 presents the designs of three Web systems enabling operation of the system on a desired level of quality:

- the WEDF system scheduling HTTP requests at the input to the system and containing only one Web server,
- the MLF system controlling a locally distributed Web cluster,
- the GGARDiB system working in the envirement of globally distributed Web clusters.

The quality of the operation of the systems presented in chapter 3 and 4 has been evaluated through simulation experiments.

The monograph has been summarized in the last Chapter 5. The plans for future research on quality-based tima-aware Web systems e.g. e-commerce, SOA and CDN systems offering differentiated quality of service have been presented.