AKADEMIA GÓRNICZO-HUTNICZA
IM. STANISŁAWA STASZICA W KRAKOWIE

WYDZIAŁ ELEKTROTECHNIKI, AUTOMATYKI, INFORMATYKI I ELEKTRONIKI
KATEDRA AUTOMATYKI

ROZPRAWA DOKTORSKA

# MODELOWANIE I ANALIZA INTERAKTYWNYCH SYSTEMÓW INTERNETOWYCH REALIZUJĄCYCH OBSŁUGĘ SZYBKOZMIENNYCH OFERT

mgr inż. Tomasz RAK

Promotor:
dr hab. inż. Jan Werewka, prof. AGH

Kraków, 2007

# Podziękowania

*Nie sposób wymienić wszystkich osób, które w najróżniejszy sposób przyczyniły się do powstania niniejszej pracy. Dziękuję im wszystkim.*

*Pragnę złożyć szczególne podziękowania Panu Profesorowi Janowi Werewce za zainspirowanie mnie do podjęcia badań naukowych, których rezultatem jest niniejsza rozprawa oraz za ogromną życzliwość i ciągłe wspieranie moich poczynań.*

*Wyrażam wdzięczność wszystkim pracownikom Katedry Informatyki i Automatyki Politechniki Rzeszowskiej, a przede wszystkim Panu Profesorowi Leszkowi Trybusowi za stworzenie przychylnych warunków do realizacji moich prac badawczych i okazaną pomoc. Doceniam również przychylność i cenne uwagi merytoryczne pracowników Katedry Automatyki Akademii Górniczo-Hutniczej w Krakowie.*

*Swoje podziękowania kieruję również do Pana dra inż. Sławomira Samoleja (Politechnika Rzeszowska) za konsultacje dotyczące modeli TCPN (Design/CPN) i bezcenną pomoc, Pana dra inż. Samuela Kouneva (University of Cambridge) za udostępnienie narzędzia do badań QPN i konsultacje dotyczące modeli kolejkowych systemów internetowych, Pana dra Matthiasa Wiesmanna (Japan Advanced Institute of Science and Technology) za udostępnienie kodów źródłowych aplikacji symulatora CSIM modelującej różne typy replikacji oraz do Pana dr inż. Krzysztofa Zatwarnickiego (Politechnika Opolska) za udostępnienie artykułów dotyczących pakietu CSIM.*

*Szczególne podziękowania kieruję do Pana dra inż. Kazimierza Lal za wspieranie moich wysiłków naukowych i nieocenioną pomoc merytoryczną.*

*Autor*

SPIS TREŚCI:

SPIS RYSUNKÓW:

SPIS TABEL:

SYMBOLE:

| | |
|---|---|
| *A* | - liczba elementów warstwy *front-end*, |
| *B* | - liczba elementów warstwy *back-end*, |
| *BD* | - baza danych, |
| *BD'* | - replikowana baza danych, |
| $b_k$ | - $k$ -ty element bazy danych, |
| $b_k'$ | - $k$ -ty element replikowanej bazy danych, |
| $C$ | - cena towaru, |
| $CS$ | - ciąg sesji, |
| $czytaj(b_k)$ | - operacja czytania danych, |
| $D_z^n$ | - *z*-ta sesja, składająca się z $n$ okresów, |
| $F$ | - rodzaj oferowanego towaru, |
| $f(t)$ | - funkcja gęstości, gęstość prawdopodobieństwa, |
| *FD* | - funkcja przypisująca przejściom dozory, |
| *FI* | - funkcja inicjalizacji, |
| FIFO_Q*2* | - kolejka FIFO modelująca element magazynujący dane warstwy *back-end*, |
| FIFO_Q*2_B* | - *B*-ta kolejka FIFO modelująca element magazynujący dane warstwy *back-end*, |
| *FK* | - funkcja kolorów, |
| *FLW* | - funkcja przypisująca łukom wyrażenia, |
| *FW* | - funkcja wag, |
| $FZ_0$ | - funkcja „oznakowania" początkowego, |
| *I* | - ilość towaru będącego przedmiotem oferty (np. liczba akcji), |
| *i* | - numer transakcji, |
| *j* | - numer operacji *i*-tej transakcji, |
| *l* | - liczba transakcji w czasie sesji, |
| *N* | - rozmiar bazy danych, |
| *O* | - oferta w systemie interaktywnym, |
| $O_G$ | - oferta giełdy, |
| $Op_i^j$ | - *j*-ta operacja *i*-tej transakcji, |
| *PACKS* | - miejsce w modelu TCPN, |
| *pBazy* | - prawdopodobieństwo opuszczenia warstwy *back-end*, |
| *pOdrzucenia* | - prawdopodobieństwo usunięcia z systemu, |
| *pReplikacji* | - prawdopodobieństwo przesłania danych do innej lokalizacji w mechanizmie replikacji, |
| *pWarstwy* | - prawdopodobieństwo opuszczenia systemu, |
| *przerwij_Tr* | - operacja przerwania transakcji, |
| *ps_quantum* | - przedział czasu obsługi dla PS (kwant), |
| PS_Q*1* | - system kolejkowy PS modelująca element przetwarzający warstwy *front-end*, |
| PS_Q*1_A* | - *A*-ty system kolejkowy PS modelująca element przetwarzający warstwy *front-end*, |
| PS_Q*2* | - system kolejkowy PS modelująca element przetwarzający warstwy *back-end*, |
| PS_Q*2_B* | - *B*-ty system kolejkowy PS modelująca element przetwarzający warstwy *back-end*, |

$Q$ - maksymalna dopuszczalna liczba zapytań w systemie kolejkowym,
$r$ - parametr rozkładu wykładniczego użyty w programie LAB Fit,
$S_B$ - stan bazy danych,
$S_{B+1}$ - kolejny stan bazy danych,
$sprawdz(b_k)$ - operacja sprawdzenia możliwości realizacji transakcji,
$T$ - przejście w modelu TCPN,
$t_{D_z}$ - czas trwania $z$-tej sesji,
$\bar{t}_o$ - średnia długość przedziału pomiędzy dwoma sąsiednimi obsługiwanymi zapytaniami,
$\bar{t}_p$ - średnia długość przedziału pomiędzy dwoma sąsiednimi przybywającymi zapytaniami,
$t_{S_B}$ - chwila czasu wystąpienia stanu bazy danych,
$t_{S_{B+1}}$ - chwila czasu wystąpienia kolejnego stanu bazy danych,
$TQ$ - tablica elementów oznaczających kolejki, które zawierają kolory lub nie,
$Tr_i$ - $i$-ta transakcja,
$t_{Op_i^j}$ - czas trwania $j$-tej operacji $i$-tej transakcji,
$t_{Tr_i}$ - czas trwania $i$-tej transakcji,
$TW$ - tablica funkcji, gdzie poszczególne elementy określają wagę opóźnienia i wykonania przejścia,
$tz_{Tr_i}$ - czas złożenia $i$-tej transakcji,
$tz_{Tr_{i+1}}$ - czas złożenia kolejnej $i+1$-ej transakcji,
$V$ - regulamin kolejki,
$X$ - typ rozkładu wejściowego strumienia zapytań,
$Y$ - typ rozkładu czasów obsługi zapytań,
$Z$ - liczba stanowisk obsługi,
$z$ - numer sesji,
$zakoncz\_Tr$ - operacja pomyślnego zakończenia realizacji danej transakcji,
$zapisz(b_k)$ - operacja zapisywania danych,
$ZC$ - zbiór cen,
$ZCC$ - zbiór chwil czasowych,
$ZK$ - zbiór inwestorów,
$Z_k$ - zapytanie klienta,
$ZL$ - zbiór łuków,
$ZM$ - zbiór miejsc,
$ZOC$ - zbiór odcinków czasu,
$ZOp$ - zbiór operacji transakcji,
$ZP$ - zbiór przejść,
$ZPW$ - zbiór papierów wartościowych,
$ZPC$ - zbiór „stempli” czasowych,
$ZT$ - zbiór kolorów (typów danych),
$ZTr$ - zbiór transakcji,
$\alpha$ - parametr rozkładu wykładniczego,
$\lambda$ - parametr rozkładu wykładniczego dla średniego natężenia strumienia zgłoszeń,

$\mu$ - parametr rozkładu wykładniczego dla średniego natężenia strumienia obsługi,
$\sigma$ - prawdopodobieństwo wejścia do kolejki,
$\rho$ - intensywność ruchu,
$\eta$ - liczba zapytań przetwarzanych w systemie,
$\varphi$ - długość kolejki, liczba zapytań w kolejce,
$\psi$ - czas obsługi, czas przebywania zapytań w systemie kolejkowym,
$\vartheta$ - czas oczekiwania w kolejce,
$\gamma_0$ - czas startu „wykonywania" sieci,
*1/A* - prawdopodobieństwo przesłania zapytania do jednego z elementów warstwy *front-end*,
*1/B* - prawdopodobieństwo przesłania zapytania do jednego z elementów warstwy *back-end*.

SKRÓTY:

ACID - Atomicity Consistency Isolation Durability,
ASP - Active Server Pages,
B2B - Business to Business,
B2C - Business to Customer,
B2P - Business to Public,
CPN - Coloured Petri Net,
CPN ML - CPN Modelling Language,
C2C - Customer to Customer,
D - Deterministic,
Design/CPN - Design/Coloured Petri Nets,
DNS - Domain Name Server,
E - ciąg zmiennych niezależnych o rozkładzie Erlanga,
FIFO - First In First Out,
GFS - Global File System,
GI - General Independent,
HA - High Availability,
HCPN - Hierarchical CPN,
HP - High Performance,
HTML - HyperText Markup Language,
HTTP - Hypertext Transfer Protocol,
H - ciąg zmiennych niezależnych o rozkładzie hiperwykładniczym,
IGA - Internetowa Giełda Akcji,
ISI - Interaktywne Systemy Internetowe,
ISIROSO - Interaktywne Systemy Internetowe Realizujące Obsługę Szybkozmiennych Ofert,
IT - Internet Technology,
JSP - Java Server Pages,
J2EE - Java 2 Platform, Enterprise Edition,
LB - Load Balancing,
LIFO - Last In First Out,

| | |
|---|---|
| LVS | - Linux Virtual Server, |
| M | - Markov, |
| MVA | - Mean Value Analysis, |
| PC | - Personal Computer, |
| PE | - Performance Engineering, |
| PHP | - PHP Hypertext Preprocessor, |
| PN | - Petri Nets, |
| PS | - Processor Sharing, |
| RAC | - Real Application Cluster, |
| QN | - Queueing Networks, |
| QoS | - Quality of Service, |
| QoWS | - Quality of Web Services, |
| QPME | - Queueing Petri net Modeling Environment, |
| QPN | - Queueing Petri Nets, |
| QPN-Tool | - Queueing Petri Net-Tool, |
| QT | - Queueing Theory, |
| RR | - Round Robin, |
| SOK | - Symulator Obciążenia Klientów, |
| SPN | - Stochastic Petri Nets, |
| SI | - Systemy Interaktywne, |
| TCPN | - Time Coloured Petri Nets, |
| UML | - Unified Modeling Language, |
| WGPW | - Warszawska Giełda Papierów Wartościowych, |
| WWW | - World Wide Web, |
| Def. | - Definicja, |
| Lis. | - Listing, |
| Tab. | - Tabela, |
| Podrozdz. | - Podrozdział, |
| Por. | - Porównaj, |
| Rozdz. | - Rozdział, |
| Rys. | - Rysunek. |

# 1 WSTĘP

Głównym przedmiotem zainteresowania niniejszej rozprawy są Interaktywne Systemy Internetowe (ISI). Są to takie systemy, które przyjmują zapytania użytkowników i odpowiadają na nie według zaimplementowanych mechanizmów. Oferują one swego rodzaju komunikację bezpośrednią, pomiędzy klientem a systemem, pozwalającą na przekazywanie informacji w obie strony. W pracy będą rozważane systemy, które prezentują klientowi dane o towarach w postaci oferty. Klient kupuje lub sprzedaje towary poprzez składanie zapytań (zleceń). W praktyce w wyniku zwiększania atrakcyjności oferowanych towarów i możliwości nabywania ich przez Internet przez stale rosnącą grupę osób, zwiększa się z upływem czasu liczba przetwarzanych przez system zapytań. Powoduje to wydłużenie czasu oczekiwania na odpowiedź. Dodatkowo w trakcie realizowania transakcji sprzedaży lub kupna, oferta systemu może ulec zmianie. W przypadku, gdy przedział czasu pomiędzy zmieniającymi się ofertami systemu jest porównywalny do przedziału czasu interakcji użytkownika (czasu przygotowania i złożenia zapytania), mówi się o systemie z szybkozmiennymi ofertami [117]. Duża liczba przybywających zapytań różnych klientów zmieniając stan systemu, powoduje w konsekwencji zmianę oferty, co może implikować dezaktualizację innych zapytań obsługiwanych w tym samym czasie przez system. Takie systemy nazwano w pracy Interaktywnymi Systemami Internetowymi Realizującymi Obsługę Szybkozmiennych Ofert (ISIROSO) [48, 71, 84, 86, 117]. Różnią się więc one w istotny sposób od typowych systemów internetowych, gdzie oferta systemu nie ulega częstym zmianom [88, 89].

W systemach klasy ISIROSO wydłużenie czasu obsługi klienta, a co za tym idzie wydłużenie czasu odpowiedzi, spowodowane jest głównie przez to, że obsługiwane sekwencyjnie zapytania klienta oczekując na wykonanie tracą aktualność. Przyczyna ta ma wpływ na wydajność całego systemu. Rozwój systemów wykorzystujących technologie WWW (ang. *World Wide Web*) rodzi potrzebę prowadzenia badań nad ich wydajnością [49, 53, 126]. Chcąc poznać funkcjonowanie systemu docelowego konieczne jest opracowanie jego modeli. Wśród współczesnych podejść stosowanych do sprawdzenia prawidłowości i poprawności konstruowanych modeli wyróżnić należy trendy, polegające na dokonywaniu porównania wyników symulacji modeli z wynikami przeprowadzonych eksperymentów na systemie referencyjnym [30, 115, 118]. Symulację stosuje się na etapie projektowania, eksploatacji lub modernizacji systemów. Technika symulacji pozwala na szybsze uzyskanie wyników niż badania nad prototypami systemów rzeczywistych. Gdy uwzględnić dodatkowo koszty, jakie trzeba ponieść na badania systemów eksperymentalnych, okazuje się, że metody badań symulacyjnych są bardzo efektywne i relatywnie tanie. Symulacja ułatwia zrozumienie funkcjonowania systemów i wykrycie cech, których często nie można zaobserwować w systemach rzeczywistych. Jej zastosowanie jest przydatne szczególnie w sytuacjach, gdy nie można „badać rzeczywistości” bo jeszcze nie istnieje, jest niedostępna albo gdy bezpośrednie działanie na przedmiot badany jest niezbyt bezpieczne lub kosztowne. Eksperymenty stosuje się w przypadku chęci potwierdzenia wyników symulacji. Pociągają one za sobą konieczność budowy eksperymentalnego środowiska laboratoryjnego co, w wielu przypadkach, jest czasochłonne i nie jest proste. Proponowana analiza, wiążąca symulację z podejściem eksperymentalnym, umożliwia badanie zachowania systemów. Metody analizy charakterystyk wydajności podczas wszystkich faz cyklu tworzenia systemów są częścią dziedziny zwanej inżynierią wydajności - PE (ang. *Performance Engineering*) [101], pomocnej w ocenie poziomu wydajności.

Możliwe jest tworzenie modeli wydajnościowych, jednak bez praktycznej weryfikacji są one mało przydatne. Konieczne jest więc opracowanie komplementarnych modeli systemu bazujących na sprawdzonych metodach analizy i zweryfikowanie ich przy pomocy eksperymentów w środowisku rzeczywistym. Takie podejście daje bazę do rozwijania kolejnych bardziej skomplikowanych i szczegółowych modeli.

## 1.1 CELE I TEZA ROZPRAWY

Popularne systemy internetowe odwiedzane są miliony razy dziennie i dlatego nie jest niczym niezwykłym, że charakteryzują się one często długimi czasami odpowiedzi, czego może doświadczyć każdy internauta. Stosuje się typowe metody skrócenia czasów odpowiedzi poprzez zwiększenie wydajności sprzętu komputerowego. Jednak dla wielu, szczególnie złożonych systemów internetowych (np. ISIROSO), brak jest gotowych rozwiązań sprawdzania wydajności, które mogłyby mieć znaczenie praktyczne [48, 114, 120]. Wprowadzenie formalnego opisu systemów powinno pozwolić na rzetelną analizę wydajności. W szczególności ważne jest, aby dostęp do zasobów i czas, po jakim otrzymuje się dane, był zadowalający dla klienta, niezależnie od liczby jednocześnie korzystających z systemu użytkowników [126].

Mając na uwadze powyższe przesłanki można sformułować następujące cele rozprawy: **zaproponowanie i przebadanie poprawności komplementarnych modeli architektur warstwowych systemów internetowych klasy ISIROSO na przykładzie giełdy, które pozwolą na opracowanie zaleceń „jak zbudować" zrównoważone[1] systemy rzeczywiste odpowiadające na wzrastający strumień zapytań**.

Istotnym problemem wydaje się wybranie odpowiedniej metody opisu systemów internetowych. W literaturze dotyczącej systemów internetowych rzadko spotyka się modele opisowe ze względu na konieczność stosowania w nich bardzo dużych uproszczeń [32].

Do grupy modeli opisowych zaliczyć można np.: opisy słowne z uwzględnieniem opisów formalnych [55, 85, 118], modele biznesowe [48] czy modele w języku UML (ang. *Unified Modeling Language*) [24, 41, 96]. Modele takie służą do opisu i klasyfikacji systemów, ale nie nadają się do prowadzenia analizy wydajnościowej i dlatego nie będą szczegółowo rozważane.

Drugie podejście obejmuje metody symulacyjne. Programowe symulatory naśladują prace systemów i zachowanie klientów internetowych [100, 115, 126]. Do modeli wydajnościowych, bo o nich mowa, zaliczyć można: modele kolejkowe [39, 87], sieci Petriego - PN (ang. *Petri Nets*) [45, 74, 94], a szczególnie modele czasowych kolorowanych sieci Petriego - TCPN (ang. *Time Coloured Petri Nets*) [84, 95, 105, 107, 114] oraz symulatory wydajności (np. CSIM [97, 126], ns-2 [146], OMNET++ [147] i inne [150]). Stosowanie programów symulacyjnych nie pozwala jednak na zamodelowanie wszystkich zjawisk zachodzących w serwerach.

Kolejna metoda polega na budowaniu prototypowych systemów zawierających serwery. Badanie pracy systemów może odbywać się z użyciem oprogramowania testowego (np. dla serwerów WWW: Surge, WebBench, SpecWeb, WebStone [162, 163]), które symuluje prace klientów wysyłając zapytania. Wadą omawianej metody, w porównaniu do dwóch poprzednich, jest wysoki koszt przeprowadzania eksperymentów. Do oceny wydajności stosuje się

[1] O zrównoważeniu systemu można mówić, jeśli długość żadnej z kolejek nie wzrasta w nieskończoność a dąży do stanu ustalonego.

najczęściej metody symulacyjne, ze względu na szerokie spektrum zastosowań, elastyczność i niskie koszty [126, 127].

Na bazie zrealizowanych przez autora prac zostaną zaprezentowane rozwiązania dotyczące modelowania wybranej klasy systemów internetowych, charakteryzujących się dużą dynamiką zmian oferty (ISIROSO). Przedstawiono dwie równoległe drogi badania systemów: modele wydajnościowe oraz eksperymenty. Budowa rzeczywistego systemu referencyjnego i analiza porównawcza parametrów wydajnościowych z opracowanymi niezależnymi modelami wydajnościowymi pozwoli na końcową weryfikację poprawności tworzonych modeli symulacyjnych. Model można uznać za poprawny, jeżeli otrzymane przy jego użyciu wyniki porównane do wyników pomiarów na rzeczywistym systemie, mieszczą się w akceptowalnych granicach błędu [71]. Na podstawie analizy modeli wydajnościowych, istnieje możliwość planowania wydajniejszych systemów rzeczywistych [54].

Doświadczenia zdobyte podczas prowadzonych badań, związanych z modelowaniem ISIROSO, doprowadziły do sformułowania następującej tezy:

- **Można jednoznacznie wskazać klasę systemów internetowych z szybkozmiennymi ofertami, dla których czas odpowiedzi wydłuża się z powodu zwiększającego się strumienia zgłoszeń i dynamicznej zmiany oferty systemu, która może powodować utratę ważności sekwencyjnie obsługiwanych zleceń znajdujących się w systemie. Dla wyszczególnionej klasy możliwe jest utworzenie podstawowych modeli wydajnościowych oraz ich weryfikacja, która pozwoli na tworzenie bardziej złożonych modeli systemów, poprzez zastosowanie architektur, bazujących na odpowiednim wydzieleniu warstw oraz przetwarzaniu równoległym, wykorzystując w szczególności replikację baz danych i przetwarzanie w węzłach mających strukturę klastrową.**

Zapytania klientów oraz odpowiedzi z serwera transportowane są przez globalną sieć Internet. Ocenia się [23], że czas przejścia odpowiedzi z serwera do klienta stanowi 20-60% czasu potrzebnego na otrzymanie danych. W prezentowanych modelach wydajnościowych ISIROSO zagadnienia wydajności sieci nie będą rozważane.

## 1.2 METODYKA BADAŃ

Niniejsza rozprawa przyjmuje za punkt wyjścia istniejące rozwiązania na temat wydajności systemów wielowarstwowych [17] przedstawionych przez Kouneva [48, 50], Menasce'a [71, 73], Williamsa [122] i Woodside'a [123]. Nowym aspektem jest użycie modeli TCPN, które pozwalają na tworzenie modeli systemu charakteryzujących się wyższym stopniem szczegółowości [9]. Przy ich użyciu możliwe jest tworzenie precyzyjnych modeli systemów, mających podobne zdolności modelowania wydajność jak kolejkowe sieci Petriego - QPN (ang. *Queueing Petri Nets*) [48] oraz większe niż sieci kolejkowe - QN (ang. *Queueing Networks*) [108] czy stochastyczne sieci Petriego - SPN (ang. *Stochastic Petri Nets*) [9, 49].

Pracę podzielono zgodnie z założeniami autora [86] na:

- modelowanie bazujące na językach formalnych,
- eksperymenty przy użyciu zbudowanego środowiska laboratoryjnego.

Wydaje się, że w przyszłości znaczenie formalnych języków projektowania i specyfikacji systemów będzie wzrastać. Współczesne języki formalne nie dorównują wprawdzie opisom stosowanym w metodykach nieformalnych, oferują jednak możliwość ścisłego wykazania

określonych właściwości opisywanych systemów. O dużym zainteresowaniu procesami zachodzącymi w systemach internetowych świadczyć mogą prace [48, 51, 114, 115] przedstawiające formalne sposoby specyfikacji. Z prac tych można wnioskować, że sieci Petriego wysokiego poziomu oferują wystarczającą „siłę ekspresji" dla modelowania i analizy złożonych systemów internetowych. Przyjęto, że językiem formalnym opisu, na bazie którego przeprowadzone zostaną badania, będą modele sieci TCPN. Sieci takie cechują się odpowiednią semantyką [45], a ponadto dostępne są narzędzia programowe (np. Design/CPN [131], CPN Tools [134]) umożliwiające konstruowanie grafów, symulację i analizę przestrzeni stanów [95]. Efektem projektowania ISIROSO będą więc czasowe kolorowane sieci Petriego, stanowiące jednocześnie dokumentację i zarazem modele systemów w postaci symulatora. Dodatkowo dla porównania i w celu poszerzenia otrzymywanych wyników wykorzystano inżynierskie narzędzie CSIM będące biblioteką funkcji do badania wydajności systemów. Tworzone przy użyciu tego środowiska programistycznego aplikacje budowane są z gotowych komponentów, umożliwiając symulację elementów składowych komputerów i łączących je sieci [23, 97, 120, 129]. Dzięki możliwości obserwacji wielu parametrów, a w szczególności długości kolejek i czasów odpowiedzi, tworzone aplikacje symulacyjne, wydają się być przydatne do badania wydajności systemów internetowych.

Eksperymenty w rzeczywistym środowisku testowym to powszechnie używany sposób do wyznaczania charakterystyk wydajnościowych [48, 50, 51, 67, 77, 84, 111, 114, 115, 119]. Narzędzia testujące „zastępują" setki a nawet tysiące „wirtualnych" użytkowników, „udających" rzeczywistych klientów. Podczas wykonywania eksperymentu mogą być monitorowane i mierzone parametry wydajnościowe (np. czas odpowiedzi). Rezultaty uzyskane taką metodą są wykorzystywane do weryfikacji modeli symulacyjnych jak również do modyfikacji systemów w celu poprawy ich wydajności. Tezę rozprawy zdecydowano się wykazać przez zaprojektowanie i przebadanie modeli wydajnościowych architektur ISIROSO oraz ich weryfikację w rzeczywistym środowisku referencyjnym. Prowadzone badania mają wykazać, że zastosowanie modelowania pozwoli na budowę poprawnych i wydajnych systemów rzeczywistych [89].

Opracowane modele będą analizowane ze względu na:

- długości kolejek,
- czasy odpowiedzi warstw systemu.

Podstawowe cechy ISIROSO obejmują następujące własności:

- szybkozmienność oferty,
- realizowanie transakcji klientów może uniemożliwić obsługę kolejnych transakcji dla innych klientów znajdujących się w systemie,
- wielokrotne powracanie zapytań do obsługi w konkretnym węźle.

Na czas realizacji prac badawczych przyjęto dodatkowo założenia:

- zastosowanie metod formalnych i nieformalnych do analizy wydajności,
- zastosowanie systemów kolejkowych do modelowania elementów warstw systemu,
- nieograniczony czas oczekiwania w kolejce,
- równomierne prawdopodobieństwo rozdzielania zapytań [*1/liczba elementów*] pomiędzy węzły w tej samej warstwie,
- równomierne prawdopodobieństwo odrzucania zapytań,
- trzy typy obciążenia 100, 300 i 500[zapytań/s],
- nieograniczoną przepustowość sieci komputerowej łączącej elementy architektury,
- ograniczenie analizy do czterech podstawowych architektur ISIROSO.

## 1.3 PLAN WYKONANYCH PRAC

W celu udowodnienia słuszności tezy zaplanowano realizację szeregu prac, których wykonanie podzielono na następujące etapy:

- przegląd rozwiązań literaturowych,
- opracowanie modelu podstawowego (model I) i jego weryfikacja,
- analiza propozycji modyfikacji wydajnościowego modelu podstawowego,
- opracowanie zmodyfikowanych modeli wydajnościowych systemu (modele II, III i IV),
- badania symulacyjne w oparciu o utworzone modele ISIROSO,
- opracowanie projektu systemu referencyjnego,
- budowa i konfiguracja środowiska laboratoryjnego dla wybranej klasy ISIROSO,
- analiza wydajności proponowanych modeli,
- weryfikacja eksperymentalna wyników symulacji,
- opracowanie wyników badań pod kątem sprawdzenia spójności założeń.

Struktura rozprawy jest konsekwencją zaproponowanych wcześniej metod i etapów wykazania tezy. Opis prac został ujęty w następujących rozdziałach:

*W rozdziale pierwszym* (Wstęp) określono zakres planowanych prac, cele i tezę rozprawy, założenia oraz sposoby rozwiązania postawionych problemów.

*W rozdziale drugim* (Modelowanie systemów internetowych) opisano znane sposoby modelowania warstwowych systemów internetowych. Przedstawiono budowę systemów internetowych. Zaprezentowano stosowane i użyteczne metody ich modelowania. Dokonano przeglądu stanu opublikowanej wiedzy w tym temacie.

*W rozdziale trzecim* (Modele ISIROSO) zawarto klasyfikację systemów e-biznesu. Przedstawiono model formalny oraz wybrane diagramy UML systemu. Na podstawie dokonanej analizy bazującej na klasyfikacji systemów e-biznesu, wybrano i uzasadniono wybór giełdy z notowaniami ciągłymi jako reprezentanta interaktywnych systemów internetowych z szybkozmiennymi ofertami. Zaprezentowano kolejkowy model tej klasy i odwzorowanie do postaci TCPN i modelu przy użyciu pakietu CSIM.

*W rozdziale czwartym* (Zagadnienie doboru architektur dla ISIROSO) omówiono sposoby zwiększania wydajności oraz rozdzielania obciążenia LB (ang. *Load Balancing*) w systemach klastrowych [28]. Przedstawiono replikację baz danych oraz podano propozycję sposobu replikacji w klastrach komputerowych. Zaproponowano zmodyfikowane modele architektur systemu giełdowego. W celu ich graficznej prezentacji użyto modeli sieci kolejkowej. Zawarto skrócony opis wykorzystanych narzędzi symulacji: Design/CPN (ang. *Design/Coloured Petri Nets*) i pakietu CSIM. Utworzono wydajnościowe modele symulacyjne TCPN i CSIM rozpatrywanych systemów.

*W rozdziale piątym* (Analiza rozwiązań) przedstawiono analizę rozwiązań badanych systemów przy wykorzystaniu modeli symulacyjnych z klastrami i replikacją bazy danych. Analizę prowadzono głownie dla czasów odpowiedzi i długości kolejek przy różnych obciążeniach ISIROSO.

*W rozdziale szóstym* (Eksperymentalna weryfikacja modeli) omówiono model biznesowy wybranego systemu z ISIROSO opisujący zachodzące procesy ekonomiczne wykorzystane w eksperymentach. Utworzono stanowisko laboratoryjne stanowiące model referencyjny ISIROSO. Rozdział zawiera głównie prezentację wyników symulacji i eksperymentów modeli z klastrami i replikacją. Weryfikacji otrzymanych wyników symulacji i eksperymentów dokonano na podstawie czasów odpowiedzi dla warstw systemu.

*W rozdziale siódmym* (*Zakończenie*) zaprezentowano uwagi i wnioski powstałe na bazie przeprowadzonych analiz: symulacji utworzonych modeli i wykonanych eksperymentów systemu referencyjnego. Zawarto propozycje dalszych prac dotyczących modelowania wydajności systemów internetowych.

Wybrane szczegóły poruszanych zagadnień, w tym między innymi opisy symulatorów, wykresy z przeprowadzonych symulacji czy opis środowiska eksperymentalnego, zostały zamieszczone w dodatku.

## 1.4 PODSUMOWANIE ROZDZIAŁU

Tematyka omawiana w niniejszej rozprawie jest dość nowa i nie doczekała się jeszcze wielu opracowań. Istnieje zaledwie kilka opublikowanych prac [48, 50, 51, 114, 115] dotyczących rozpatrywanych systemów i skupiających się na różnych sposobach modelowania pozwalających na poprawę parametrów wydajnościowych. Proponowane w literaturze sposoby poprawy wydajności nie obejmują całych systemów, a w szczególności nie podejmują problemu implementacji dla konkretnej klasy systemów internetowych.

Niniejsza rozprawa skupia się zasadniczo na modelowaniu ISIROSO w celu wskazania sposobów, pozwalających na skrócenie czasu potrzebnego na odpowiedź poprzez modyfikację architektury i parametrów systemów. Dzięki wykorzystaniu modeli możliwa jest dogłębna analiza zachowania systemów rzeczywistych bez konieczności ich budowy, co ogranicza koszty. Zastosowanie takiego podejścia pozwala na wczesne wykrycie wad w budowie systemów. W celu potwierdzenia poprawności modeli symulacyjnych, konieczne jest zastosowanie weryfikacji eksperymentalnej. Zaletą omawianego podejścia wydaje się być wykorzystanie znanych, może jeszcze nie w pełni docenionych, mechanizmów jakimi są klastrowanie i replikacja danych, które pozwalają na modyfikacje architektury warstwowych systemów internetowych.

Częściowe rezultaty prac autora rozprawy dotyczące modelowania wydajności systemów internetowych zostały opublikowane wcześniej w artykułach [58, 59, 63, 83, 84, 86, 87, 88, 89, 90, 117] a zagadnienia pokrewne w publikacjach [64, 82, 85]. Prezentowana praca badawcza finansowana jest ze środków Komitetu Badań Naukowych w latach 2006-2007 jako projekt badawczy (grant promotorski 3 T11C 022 30).

# 2 MODELOWANIE SYSTEMÓW INTERNETOWYCH

W rozdziale opisano sposoby modelowania systemów internetowych. Omówiono warstwową budowę i zasady działania tych systemów. Dokonano przeglądu modeli przydatnych do ich opisu. Modele kolejkowe i modele UML używane są do prezentacji ogólnej, natomiast modele sieci Petriego i symulator CSIM wykorzystywano do badania wydajności. W podsumowaniu rozdziału zaprezentowano przegląd literatury, który umożliwia umiejscowienie prezentowanych badań w kontekście stanu wiedzy nad systemami internetowymi.

## 2.1 PROJEKTOWANIE SYSTEMÓW INTERNETOWYCH

Dziś wiele środowisk handlu elektronicznego bazuje na wielowarstwowych i rozproszonych strukturach [49]. Złożoność takiej architektury sprawia problemy działaniu systemów tak, aby dostarczyć pożądany poziom usług - QoS (ang. *Quality of Service*) [90]. Dotyczy to szczególnie problemów:

- wystąpienia chwilowego wysokiego obciążenia, czyli dużej liczby obsługiwanych zapytań na sekundę [zapytań/s] w czasie „szczytu” (problem nadążenia z obsługą),
- wydłużenia średniego czasu odpowiedzi (zniecierpliwienie użytkowników zbyt długimi czasami odpowiedzi systemów),
- przeciążenia zasobów (spadek wydajności systemów).

Z drugiej strony zarządzający systemami internetowymi potrzebują znać odpowiedzi na następujące pytania:

- Co zrobić, aby zagwarantować odpowiednią wydajność przy wzrastającym obciążeniu?
- Jaki jest czas odpowiedzi?
- W jakim stopniu wykorzystywane są zasoby?

W większości przypadków wydajność systemów jest trudna do określenia. Brak wcześniejszych analiz przy projektowaniu może prowadzić do nieoczekiwanych problemów w działaniu systemów. Projektanci muszą określić kwestie planowania wydajności w formalny i usystematyzowany sposób. Większość z nich zazwyczaj polegają na swojej intuicji, doświadczeniu oraz ogólnych zasadach bazujących na praktyce inżynierskiej [73].

Są dwie podstawowe metody poszukiwania rozwiązań pojawiających się problemów, bazujące na PE [101, 149]:

- modelowanie systemów,
- praktyczne eksperymenty na zbudowanych systemach.

W praktyce powinny być one używane bardziej jako uzupełniające się metody, aniżeli alternatywne [53].

### 2.1.1 MODELOWANIE SYSTEMÓW

Generalnie można określić dwie główne drogi modelowania: analityczne i symulacyjne. Modele analityczne bazują na zbiorze wzorów i algorytmów. Wykorzystywane są do generowania parametrów wyjściowych (np. wydajności) ze zbioru parametrów wejściowych (np. charakterystyki obciążenia systemów). Tego typu modele bazują np. na teorii sieci kolejkowych (QN). Zasadniczo, analityczny model wydajności jest bardzo skomplikowany i wymaga szczegółowych informacji o komponentach systemu. Modele symulacyjne są programami komputerowymi, które imitują zachowanie systemów, gdy zapytania przepływają przez różne zasoby systemowe. Struktura programu bazuje na stanie symulowanych systemów i zdarzeniach, powodujących zmiany. Symulatory mogą mierzyć wydajność systemów poprzez zliczanie zdarzeń i czasów ich trwania. Główną korzyścią stosowania modeli symulacyjnych jest ich wszechstronność. Są one jednym z podstawowych sposobów prowadzenia badań w czasie konstruowania systemów [126].

Modelami użytymi do opisu systemów branymi pod uwagę w niniejszej rozprawie są:

- modele matematyczne,
- modele w języku UML [19, 161],
- modele kolejkowe [20, 33, 39, 40, 44, 65, 139].

Modelami symulacyjnymi są:

- modele wydajnościowe TCPN [1, 45, 94, 95, 106, 107,],
- modele wydajnościowe bazujące na bibliotekach oprogramowania CSIM [23, 97].

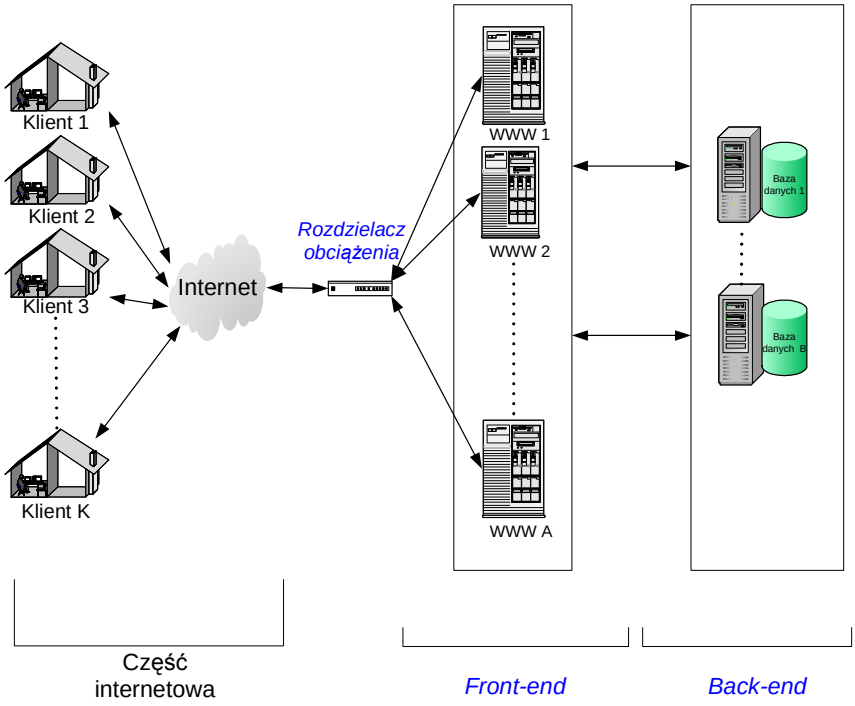
Tworzone modele wydajnościowe uważa się za miarodajne [71], jeżeli zmierzone wartości parametrów modeli nie odbiegają (maksymalnie 30% [73]) od wartości pochodzących z rzeczywistych systemów.

### 2.1.2 EKSPERYMENTY SYSTEMÓW RZECZYWISTYCH

Do określenia wydajności często używane są rezultaty badań systemów rzeczywistych. Z drugiej strony, testy obciążeniowe możliwe są dopiero wówczas, gdy systemy taki zostaną zbudowane, wymagają więc sporego nakładu czasu pracy. Kolejną wadą tego rozwiązania jest fakt, że eksperymenty odbywają się w wyizolowanym środowisku testowym, które z zasady jest różne od rzeczywistego. Jednak zastosowanie eksperymentów w połączeniu z modelowaniem ma niebagatelny wpływ na poprawność tworzonych systemów. Zbieżność charakterystyk symulacyjnych modelu z charakterystykami systemów wzorcowych jest potwierdzeniem poprawności przyjętych metod i sposobów modelowania [73].

## 2.2 WARSTWY SYSTEMÓW INTERNETOWYCH

Typowa architektura systemów internetowych składa się z przeważnie z kilku warstw. Prezentowana architektura (Rys. 2.1) jest bardzo popularna w Internecie. Poniżej zaprezentowano wykorzystywane w systemach internetowych warstwy: *front-end* i *back-end* [78].




Rys. 2.1 Warstwy w systemach internetowych

### 2.2.1 WARSTWA FRONT-END

Warstwa *front-end* (prezentacyjno-aplikacyjna) bazuje z jednej strony na serwerze WWW i np. języku HTML (ang. *HyperText Markup Language*) oraz z drugiej na serwerze aplikacji. Zadaniem serwera prezentacji jest przedstawianie klientowi oferty systemów. Zadaniem serwera aplikacji jest zarządzanie transakcjami. Nowoczesne serwery aplikacji dostarczają funkcjonalności klastrowania, która umożliwia rozdzielanie obciążenia - LB (ang. *Load Balancing*). W przypadku awarii któregoś z serwerów, zapytanie może być automatycznie i w sposób niezauważalny dla klienta przekazywane do innego węzła. W architekturze systemów występuje więc wiele serwerów, które mają zapewnić większą wydajność tej warstwy. Często, w przypadku złożonych systemów, do rozdzielania obciążenia na poszczególne serwery wykorzystane są urządzenia sieciowe równoważące obciążenie [6] czy usługi sieciowe pozwalające rozdzielać ruch (np. DNS (ang. *Domain Name Server*)). Systemy łączące, w jednej warstwie, te dwie funkcje: prezentację i zarządzanie, wykorzystują technologie np. ASP (ang. *Active Server Pages*), JSP (ang. *Java Server Pages*) czy PHP (ang. *PHP Hypertext Preprocessor*).

### 2.2.2 WARSTWA BACK-END

Warstwa *back-end* (danych) zawiera jedną lub w przypadku replikacji kilka baz danych. Warstwa ta przechowuje dane systemów, a dostęp do nich realizowany jest poprzez transakcje. Wykorzystanie mechanizmu klastrowania i replikacji pozwala na zrównoleglenie obsługi zapytań, a dodatkowo na zwiększenie bezpieczeństwa przechowywanych danych. Większość baz danych zawiera mechanizmy replikacji danych na różnych poziomach złożoności.

# 2.3 METODY MODELOWANIA

W podrozdziale opisano uznane i stosowane metody modelowania systemów internetowych. Możliwość ich zastosowania została potwierdzona w wielu publikacjach naukowych na ten temat.

## 2.3.1 JĘZYK UML

UML jest standardem wśród języków specyfikujących, wizualizujących, konstruujących oraz dokumentujących systemy i aplikacje komputerowe. Został użyty w licznych projektach modelujących architektury i wymagania systemów internetowych (np. [7, 24, 41, 93]).

Głównym środkiem wyrazu języka UML jest zestaw diagramów przedstawionych w postaci grafów, których wierzchołkami są elementy, a krawędziami związki pomiędzy nimi. Notacja UML proponuje kilka rodzajów diagramów [19]. Są to między innymi diagramy:

- Przypadków użycia (ang. *use case*) - opisujące, „co robią" systemy z punktu widzenia zewnętrznego obserwatora. Użyty w UML-u aktor to ktoś albo coś, co inicjuje zdarzenia.
- Klas (ang. *class*) - przedstawiające ogólną „panoramę" systemów, pokazując klasy i ich wzajemne relacje. Diagramy te mają one charakter statyczny - pokazują, co wchodzi w interakcje, a nie co się dzieje podczas tych interakcji.
- Obiektów (ang. *objects*) - które w zastępstwie klas prezentują instancje. Używane są do statycznego opisywania elementów ze złożonymi relacjami, zwłaszcza rekurencyjnymi.
- Sekwencji (ang. *sequence*) - które szczegółowo pokazują, w jaki sposób są wykonywane operacje. Są to diagramy dynamiczne opisujące jak obiekty ze sobą współpracują.
- Maszyny stanów (ang. *statechart*) - pokazują możliwe stany obiektu oraz przejścia między nimi.
- Czynności (ang. *activity*) - czyli aktywności, to szczególny przypadek diagramów stanu. Obrazują one strumień kolejno wykonywanych czynności. Odnosi się to do modelowania dynamicznych aspektów systemów. Na diagramach czynności można także zobrazować zmiany zachodzące w obiekcie, gdy przechodzi on z jednego stanu do drugiego.
- Wdrożenia (ang. *deployment*) - pokazujące fizyczną konfigurację oprogramowania i sprzętu.

## 2.3.2 SIECI KOLEJKOWE

Często stosowaną grupą są modele odtwarzające zdarzenia zachodzące w sieci kolejek. Modele takie, mimo licznych uproszczeń, potrafią dobrze uchwycić zależności pomiędzy parametrami systemów. Sieć kolejkowa w aspekcie modelowania pracy systemów komputerowych jest siecią wzajemnie połączonych systemów kolejkowych, które reprezentują systemy komputerowe [34]. Teoria kolejek - QT (ang. *Queueing Theory*), zwana także teorią masowej obsługi, zajmuje się modelowaniem i optymalizacją wszelkiego rodzaju stanowisk obsługi. Głównym zadaniem teorii kolejek jest minimalizacja czasu i kosztów obsługi. Oczekiwanie w kolejce jest czasem marnotrawionym, dlatego też redukcja długości czasu

oczekiwania poprzez zwiększenie liczby stanowisk obsługi w systemach może przynieść zwiększenie korzyści. Z drugiej zaś strony, nadmierne zwielokrotnienie stanowisk obsługi może doprowadzić do niedociążenia stanowiska obsługi i strat finansowych. Zasadniczym celem teorii masowej obsługi jest opracowanie ogólnych metod, pozwalających wyznaczyć podstawowe wskaźniki procesu obsługi i umożliwiających ocenę jakości stanowisk obsługi oraz wybór jego struktury i przepływów. Z punktu widzenia użytkownika systemów należy wypracować wskazania dla podjęcia decyzji o sposobie użytkowania systemów, natomiast z punktu widzenia zarządzającego systemami, należy określić warunki najbardziej efektywnego ich wykorzystania.

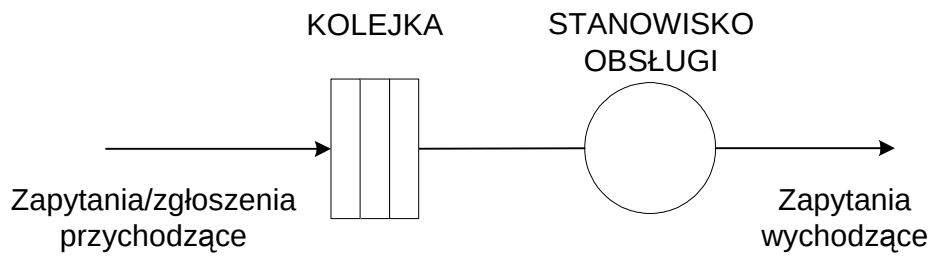
Sieci kolejkowe zgrupowane są w dwóch kategoriach:

- Otwarte, jeśli jest możliwe przybycie zapytań z zewnątrz i opuszczenie sieci. Jest to opisywane poprzez wprowadzenie fikcyjnego węzła *0* reprezentującego otoczenie.
- Zamknięte, jeśli zewnętrzne przybycia i odejścia zapytań są niedozwolone, czyli maksymalna liczba zapytań w systemie jest ustalona.

Sieci kolejkowe są bardzo popularne dla oceny wydajności poprzez analizę ilościową. Aby dokonać analizy dowolnego systemu kolejkowego konieczne jest określenie:

- procesu przybyć/wejścia zapytań do systemu - czyli rozkładu prawdopodobieństwa odstępów pomiędzy kolejnymi zgłoszeniami do systemu,
- rozkładu prawdopodobieństwa czasów obsługi zapytań,
- regulaminu kolejki - sposobu, w jaki kolejne zapytania są wybierane do obsługi,
- możliwość przebywania w kolejce - polegającej na tym czy napływające zapytania mają możliwość oczekiwania w kolejce lub sposobu odrzucania w przypadku, gdy wszystkie urządzenia obsługujące są zajęte - określenie maksymalnej długości kolejki [40].

Tworzenie sieci kolejkowych polega na ustaleniu połączeń poszczególnych elementów struktury [102]. Każdy element jest systemem kolejkowym (Rys. 2.2), zawierającym podelement przechowujący (kolejka), gdzie przybywają zapytania (według określonego rozkładu prawdopodobieństwa) i oczekują na obsługę oraz podelement realizacji zapytań (stanowisko obsługi), gdzie zapytania trafiają według określonego regulaminu kolejki i są obsługiwane. Każde zapytanie żąda określonego czasu obsługi, który wyznacza przedział czasu, jaki zajmuje ono w stanowisku obsługi. Dla przybywających zapytań obsługa zostanie rozpoczęta natychmiast, jeśli jest wolne stanowisko obsługi. W przeciwnym razie zapytania zmuszone są oczekiwać w kolejce [110].




Rys. 2.2 System kolejkowy

System kolejkowy powinien obsługiwać zgłaszające się zapytania z szybkością większą niż ich przybywanie. Istnieją przedziały czasowe, w których przybywa więcej zgłoszeń niż może być jednocześnie obsłużonych i część z nich (o ile jest to możliwe) musi czekać na

obsługę. Zapytania te tworzą kolejkę. Tak zdefiniowany system kolejkowy można opisać za pomocą [39]:

- strumienia zgłoszeń - czyli statystycznego opisu procesu przybywających do systemu zapytań (z intensywnością napływania $\lambda$ - średnia liczba zapytań nadchodzących w jednostce czasu, czyli natężenie strumienia zgłoszeń [liczba zapytań/jednostkę czasu]),
- procesu obsługi - realizującego obsługę zapytań (z intensywnością obsługi $\mu$ - średnia liczba zapytań obsługiwanych w systemie w przedziale czasu, czyli natężenie obsługi [liczba zapytań obsłużonych/jednostkę czasu]),
- regulaminu kolejki - czyli metod wybierania następnego zapytania do obsługi, w przypadku istnienia kolejki.

Strumień zgłoszeń jest opisywany za pomocą funkcji rozkładu odstępów czasu (interwałów) między kolejnymi zapytaniami. Gdy zapytania są losowe, interwał jest zmienną losową i można określić jego funkcję rozkładu prawdopodobieństwa (gęstość prawdopodobieństwa) [39]. Najbardziej popularnym i najczęściej stosowanym rozkładem prawdopodobieństw, do modelowania procesu napływania i obsługi zapytań internetowych, jest rozkład wykładniczy [39], którego funkcja gęstości (gęstość prawdopodobieństwa) ma następującą ogólną postać:

$$f(t) = \alpha e^{-\alpha t} \qquad dla \qquad t \geq 0 \tag{2.1}$$

gdzie $\alpha$ jest parametrem rozkładu wykładniczego.

Przyjmując jako $\bar{t}_p$ - średnią długość interwału pomiędzy dwoma sąsiednimi przybywającymi zapytaniami, parametr rozkładu wykładniczego dla przybywających zgłoszeń $\lambda$ można opisać jako:

$$\alpha = \lambda = \frac{1}{\bar{t}_p} \tag{2.2}$$

Jeżeli czas pomiędzy dwoma przybywającymi zapytaniami wynosi średnio $\bar{t}_p = \frac{1}{5}[s]$, co oznacza obciążenie 5[zapytań/s], wtedy $\lambda = 5[1/s]$.

Często zdarza się, iż czas obsługi zapytania w systemie nie jest stały. Jeżeli podlega on stochastycznym wahaniom, to zazwyczaj jest opisywane za pomocą odpowiedniej funkcji rozkładu, najczęściej wykładniczego [48, 115]. Przyjmując jako $\bar{t}_o$ - średnią długość interwału pomiędzy dwoma sąsiednimi obsługiwanymi zapytaniami, parametr rozkładu wykładniczego można zapisać jako:

$$\alpha = \mu = \frac{1}{\bar{t}_o} \tag{2.3}$$

Jeżeli czas pomiędzy dwoma obsłużonymi zapytaniami wynosi średnio $\bar{t}_o = \frac{1}{6}[s]$, co oznacza obsługę 6[zapytań/s], wtedy $\mu = 6[1/s]$.

Regulamin kolejki jest trzecim elementem wpływającym na stan systemu kolejkowego i określającym kolejność wybierania zapytań z kolejki. Podstawowe regulaminy kolejki stosowane do modelowania systemów internetowych to:

- FIFO (ang. *First In First Out*) - jako pierwsze do obsługi kieruje się zapytanie najdłużej oczekujące w kolejce.
- LIFO (ang. *Last In First Out*) - jako pierwsze do obsługi kieruje się zapytanie najkrócej oczekujące w kolejce.
- RR (ang. *Round Robin*) - traktuje zapytania według regulaminu FIFO, lecz obsługa jest przerywana na końcu określonego przedziału czasu zwanego kwantem ( $ps\_quantum$ ). Jeśli obsługa nie została zakończona przed upływem tego czasu, zapytanie zajmuje miejsce w kolejce z prawdopodobieństwem $\sigma$ lub opuszcza system w przypadku zakończenia obsługi z prawdopodobieństwem $(1-\sigma)$.
- PS (ang. *Processor Sharing*) - reprezentuje graniczny przypadek RR, gdy przedział czasu obsługi jest bardzo mały.

Podstawowym oznaczeniem dla kolejek jest zapis *X/Y/Z* zaproponowany przez D. Kendalla, rozszerzony w notacji A. M. Lee - *X/Y/Z/V/Q*. W notacji tej kolejne symbole oznaczają [39]:

- *X* - typ rozkładu wejściowego strumienia zapytań,
- *Y* - typ rozkładu czasów obsługi zapytań,
- *Z* - liczbę stanowisk obsługujących ( $Z \geq 1$ ),
- *V* - regulamin kolejki,
- *Q* - rozmiar systemu, czyli maksymalną liczbę zapytań w kolejce.

Typy rozkładów to:

- *GI* (ang. *General Independent*) - oznacza, że odstępy pomiędzy przybywającymi zapytaniami stanowią ciąg zmiennych losowych niezależnych o jednakowym rozkładzie,
- *M* (Markov) - oznacza, że odstępy pomiędzy przybywającymi zapytaniami stanowią ciąg zmiennych losowych niezależnych o rozkładzie wykładniczym,
- *D* (ang. *Deterministic*) - oznacza, że odstępy pomiędzy przybywającymi zapytaniami mają jednakową i stałą wartość,
- *E* - oznacza, że odstępy pomiędzy przybywającymi zapytaniami stanowią ciąg zmiennych niezależnych o rozkładzie Erlanga,
- *H* - oznacza, że odstępy pomiędzy przybywającymi zapytaniami stanowią ciąg zmiennych niezależnych o rozkładzie hiperwykładniczym.

Średnie wartości parametrów systemów kolejkowych można wyznaczyć w sposób analityczny (Tab. 2.1).

Tab. 2.1 Wzory analityczne do wyznaczania parametrów systemów kolejkowych FIFO i PS [39]

| Parametr \ Kolejka | M/M/1/FIFO/∞ | M/M/1/PS/∞ |
|---|---|---|
| Intensywność ruchu | $\rho = \lambda / \mu$ | $\rho = \lambda / \mu$ |
| Średnia liczba zapytań w systemie | $\overline{\eta} = \frac{\rho}{1-\rho} = \frac{\lambda}{\mu - \lambda}$ | $\overline{\eta} = \frac{\rho}{1-\rho} = \frac{\lambda}{\mu - \lambda}$ |
| Średni czas obsługi | $\overline{\psi} = \frac{\overline{\eta}}{\lambda} = \frac{1}{\mu - \lambda}$ | $\overline{\psi} = \frac{1/\mu}{1-\rho} = \frac{\mu}{\mu - \lambda}$ |
| Średnia długość kolejki | $\overline{\varphi} = \frac{\rho^2}{1-\rho}$ | $\overline{\varphi} = \frac{\rho^2}{1-\rho}$ |
| Średni czas oczekiwania w kolejce | $\overline{\vartheta} = \frac{\overline{\varphi}}{\lambda} = \frac{\lambda}{\mu(\mu - \lambda)}$ | $\overline{\vartheta} = \frac{\overline{\varphi}}{\lambda} = \frac{\lambda}{\mu(\mu - \lambda)}$ |

Powyższy opis sieci kolejkowych przedstawiono w celu przybliżenia zasad budowy systemów kolejkowych, które w dalszej części wykorzystano jako podstawę do konstrukcji modeli wydajnościowych.

### 2.3.3 SIECI PETRIEGO

Sieci Petriego (PN) [74, 79] szczególnie nadają się do specyfikowania i analizy systemów współbieżnych. Dynamika systemów opisywanych przy pomocy sieci odzwierciedlana jest przez znaczniki (ang. *tokens*) przemieszczające się w grafie sieci. Graf sieci można poddać formalnej analizie, dlatego sieci Petriego określane są jako połączenie pomiędzy inżynierskim sposobem opisu systemów w postaci diagramu a podejściem teoretycznym, opisującym model w postaci wzorów matematycznych [74].

Sieć Petriego składa się z miejsc (ang. *places*), przejść (ang. *transitions*) oraz łuków (ang. *arcs*). Łuki łączą miejsca z przejściami i na odwrót. Między dwoma miejscami oraz między dwoma przejściami nie mogą występować łuki. Miejsce może zawierać dowolną liczbę znaczników. Przejścia pełnią funkcje pobierania znaczników z pozycji wejściowych i przekazywania ich na pozycje wyjściowe. Przejście jest aktywne, gdy na każdej z pozycji wejściowych istnieją jeszcze znaczniki. Miejsca sieci interpretuje się zazwyczaj jako warunki, stany lub zasoby, przejścia jako akcje, a znaczniki to zapytania. Struktura sieci Petriego odwzorowuje wtedy budowę systemów, a ruch znaczników w sieci modeluje postępujące wykonanie procesów przetwarzanych zapytań [104, 112].

**Definicja 2.1.**

Sieć Petriego jest krotką $PN = (ZM, ZP, ZL, FW, FZ_0)$, gdzie [74]:

- $ZM$ - jest skończonym zbiorem miejsc; $ZM = \{mm_1, mm_2, ..., mm_a\}$,
- $ZP$ - jest skończonym zbiorem przejść; $ZP = \{pp_1, pp_2, ..., pp_b\}$,
- $ZL$ - jest zbiorem łuków; $ZL \subseteq ZM \times ZP \cup ZP \times ZM$,

- $FW$ - jest funkcją wag przypisującą etykiety do każdego łuku; $FW : ZL \rightarrow \{1,2,...\}$,
- $FZ_0$ - jest funkcją, tzw. „oznakowaniem" początkowym (ang. *initial marking*), odwzorowującą zbiór miejsc w zbiór nieujemnych liczb całkowitych i wiążącą jednocześnie znaczniki z miejscami sieci; $FZ_0 : ZM \rightarrow \{0,1,...\}$.

Zakłada się ponadto, że dla każdej sieci zbiory miejsc i przejść są rozłączne $ZM \cap ZP = \varnothing$ i $ZM \cup ZP \neq \varnothing$.

Sieci Petriego, podobnie jak sieci kolejkowe, są dobrze znanymi modelami służącymi do opisywania i analizowania stanowisk obsługi. Sieci Petriego służą do analizy jakościowej działania systemów, polegającej na określeniu poprawności logicznej. Dla analizy ilościowej, polegającej na określaniu efektywności, sieci Petriego nie mogą być stosowane ze względu na brak w nich aspektu czasu. Niektóre odmiany PN, takie jak stochastyczne sieci Petriego [52] lub czasowe sieci Petriego, próbują sprostać wymaganiom analizy ilościowej.

W literaturze napotkać można prace mówiące o badaniu wydajności serwisów WWW z wykorzystaniem sieci Petriego. Badania skupiają się na badaniu obciążenia systemów [8], pomiarach np. czasu reakcji [2] lub prezentacji ogólnego planu modelowania [36, 115].

KOLOROWANE SIECI PETRIEGO

Klasyczne sieci Petriego służą do modelowania przepływu sterowania, pozbawionego możliwości opisu przetwarzania danych i wpływu wyników tego przetwarzania na przebieg procesu obliczeniowego. Kolorowane sieci Petriego CPN (ang. *Coloured Petri Net*), jako język formalnego opisu systemów, oprócz ścisłej semantyki oferują rozróżnianie znaczników i możliwość ich definiowania w postaci struktur danych.

CPN rozszerza model wprowadzając do niego wartości przypisane do „krążących" w sieci znaczników oraz funkcje sterujące przepływem tych znaczników i obliczające nowe wartości. Kolorowana sieć Petriego jest grafem złożonym z miejsc, przejść i łuków, w którym poruszają się znaczniki przenoszące wartości określonego typu. W każdym miejscu sieci może znajdować się wiele różnych znaczników, których typ musi być jednak zgodny z typem tego miejsca. „Wzbudzenie" przejścia zależy zarówno od obecności znaczników w miejscach wejściowych przejścia, jak i od wartości przypisanych do tych znaczników. „Odpalenie" przejścia powoduje usunięcie pewnej liczby znaczników z miejsc wejściowych przejścia i utworzenie pewnej liczby nowych znaczników w miejscach wyjściowych [45]. Z miejscem związany jest typ danych (zbór kolorów). Stan sieci CPN nazywa się „oznakowaniem", a oznacza on liczbę znaczników rozmieszczonych w miejscach. Znaczniki są nośnikami określonej wartości danych (koloru) należących do typu powiązanego z miejscem. Miejsca i przejścia mogą być łączone przy użyciu większej liczby łuków nazywanych zbiorem łuków. Łuki posiadają wyrażenia (ang. *expressions*) określające ilość i wartość przesyłanych danych. Obok wyrażeń łukowych na zachowanie sieci mają wpływ logiczne wyrażenia, powiązane z przejściami, nazywane dozorami (ang. *guards*).

**Definicja 2.2.**
Kolorowana sieć Petriego jest uporządkowaną krotką
$CPN = (ZT, ZM, ZP, ZL, FW, FK, FD, FLW, FI)$, gdzie [45]:

- $ZT$ - jest skończonym zbiorem kolorów (typów danych),
- $ZM$ - jest skończonym zbiorem miejsc,
- $ZP$ - jest skończonym zbiorem przejść,
- $ZL$ - jest skończonym zbiorem łuków;
  $ZM \cap ZP = ZM \cap ZL = ZP \cap ZL = \varnothing$,
- $FW$ - jest funkcją wierzchołków; $FW : ZL \rightarrow ZM \times ZP \cup ZP \times ZM$,
- $FK$ - jest funkcją kolorów przypisującą miejscom palety kolorów; $FK : ZM \rightarrow ZT$,
- $FD$ - jest funkcją przypisującą przejściom dozory, których spełnienie powoduje akceptację elementu,
- $FLW$ - jest funkcją przypisującą łukom wyrażenia,
- $FI$ - jest funkcją inicjalizacji.

Funkcje $FLW$ i $FI$ operują na pojęciu wielozbioru czyli istnienia w więcej niż jednego egzemplarza pojedynczego elementu w zbiorze, a liczbę wystąpień określa się poprzez zwiększenie odpowiedniego współczynnika.

Złożone modele CPN tworzy się przy użyciu hierarchicznych kolorowanych sieci Petriego HCPN (ang. *Hierarchical CPN*), poprzez łączenie ze sobą mniejszych sieci według określonych zasad. Najczęściej w postać podstrony (podsieci CPN) przenoszone jest przejście wraz z otaczającymi go miejscami i łukami. Nazywane jest ono przejściem „podstawianym" (ang. *substitution transition*). Odzwierciedleniem relacji pomiędzy poszczególnymi stronami jest graf hierarchii sieci (ang. *page hierarchy graph*) [45, 107].

CZASOWE KOLOROWANE SIECI PETRIEGO

Czasowe kolorowane sieci Petriego (TCPN) to rozszerzenie CPN o globalny zegar, którego stan utożsamiany jest ze stanem modelowego czasu. Pewna pula znaczników otrzymuje dodatkową wartość zwaną „stemplem" czasowym (ang. *time stamp*), której wartość dodatkowo determinuje wykonanie przejścia.

**Definicja 2.3.**
Czasowa kolorowana sieć Petriego jest uporządkowaną krotką
$TCPN = (CPN, ZPC, \gamma_0)$, gdzie [45]:

- $CPN$ - jest zgodne z definicją 2.2,
- $ZPC$ - jest zbiorem „stempli" czasowych,
- $\gamma_0$ - jest czasem startu „wykonywania" sieci i $\gamma_0 \in R$.

Szczegółowe informacje dotyczące TCPN można znaleźć w [45, 105]. Dla tych sieci opracowano różne narzędzia analizy [131, 134]. Często stosowanym jest pakiet Design/CPN [131], a ze względu na posiadany moduł analizy wydajnościowej (moduł *Performance*) [116] można również zastosować go do rozpatrywanego w tej rozprawie problemu.

KOLEJKOWE SIECI PETRIEGO

QPN, czyli kolejkowe sieci Petriego [12, 13], posiadają duże możliwości modelowania dzięki połączeniu właściwości sieci kolejkowych i sieci Petriego. Proces modelowania polega na tym, że najpierw należy opisać logiczną strukturę systemów używając podstawowych elementów sieci Petriego, a następnie zadaje się wartości parametrów.

Wybrane miejsca czasowe zawierają kolejkę dla obsługiwanych znaczników. Znaczniki dodawane do miejsc, po „odpalonej" tranzycji, są umieszczane w kolejce według harmonogramu kolejki. Po obsłudze odpowiedni znacznik przechodzi do kolejki, gdzie jest udostępniony dla tranzycji wyjściowej miejsca. Znacznik w kolejce nie jest udostępniony do „odpalenia". Każdy model QPN może być analizowany pod względem jakościowym i ilościowym. Integracja węzłów podsieci w pierwotny formalizm QPN prowadzi do hierarchicznej struktury modelu QPN.

**Definicja 2.4.**

Kolejkowa sieć Petriego to krotka $QPN = (CPN, TQ, TW)$, gdzie [48]:

- $CPN$ - to kolorowana sieć Petriego,
- $TQ$ - jest tablicą gdzie poszczególne elementy oznaczają kolejki zawierające kolory lub ich brak, w zależności od rodzaju miejsca czasowego lub nie czasowego; $TQ = (qq_1, ..., qq_{|ZM|})$,
- $TW$ - jest tablicą funkcji gdzie poszczególne elementy (wagi) określają opóźnienie wykonania przejścia spowodowane typem danych (kolorem), dla przejść czasowych lub natychmiastowych; $TW = (ww_1, ..., ww_{|ZP|})$.

Istniejące narzędzia analizy QPN to QPME (ang. *Queueing Petri net Modeling Environment*) [153] oraz QPN-Tool [154].

### 2.3.4 PAKIET CSIM

W celu praktycznego badania modeli konieczne jest wykorzystanie komputera, a kolejnym logicznym krokiem jest przygotowanie bibliotek dla programów symulacyjnych. Opracowanie takich bibliotek sprawia, że modelowaniem mogą zajmować się nie tylko specjaliści w tej dziedzinie. Do najbardziej znanych pakietów tego typu należą między innymi: darmowe pakiety ns-2 [146], OMNET++ [147] i inne [150] oraz komercyjny pakiet CSIM firmy Mesquite [130].

Pakietem najczęściej wykorzystywanym, z punktu widzenia modelowania systemów internetowych, jest CSIM [97, 119, 120, 121, 126, 130]. Charakteryzuje się on dużą przejrzystością i zróżnicowaniem zastosowanych funkcji i obiektów. Umożliwia, w prosty i łatwy do zrozumienia sposób, generowanie w języku C lub C++ bardzo złożonych modeli symulacyjnych. CSIM jest więc biblioteką funkcji. Narzędzie to pozwala na budowanie modeli systemów o dowolnej liczbie elementów i różnej strukturze. CSIM okazuje się szczególnie przydatny podczas projektowania przykładowo różnego rodzaju systemów komputerowych [97]. W CSIM wyróżnia się następujące typy obiektów:

- *Process* - używany do modelowania zapytań, klientów i serwerów oraz innych aktywnych składników systemów związanych z generowaniem ruchu.

- *Facility* - używany do modelowania zasobów, z jedną kolejką zapytań oraz jednym lub wieloma stanowiskami obsługi, które są wykorzystywane przez procesy.
- *Storage* - używany do modelowania zasobów, które mogą być w całości lub częściowo przydzielone procesom. Obiekty tego typu posiadają licznik elementów określający dostępną „pojemność" danego zasobu. Posiadają również kolejkę oczekujących procesów. Proces może zająć zasób, a w przypadku braku miejsca zawiesza prace i pozostaje w kolejce w oczekiwaniu na elementy zasobu.
- *Buffer* - zasób przechowujący zapytania, używany do modelowania kolejek o skończonej pojemności. Może częściowo zostać przydzielony procesowi, posiada licznik miejsc oraz dwie kolejki: kolejkę procesów oczekujących na wolne miejsce oraz kolejkę procesów oczekujących na wyjście. Zasoby tego typu doskonale nadają się do modelowania zapytań klientów.
- *Event* - zdarzenia używane do synchronizacji i kontroli interakcji między procesami. Jeden proces może wygenerować zdarzenie, inny może oczekiwać na zdarzenie. Kilka procesów oczekujących na to samo zdarzenie może zostać ustawionych w kolejkę. Proces będąc w kolejce może być „zawieszony" w oczekiwaniu na zdarzenie i po przekroczeniu czasu oczekiwania wznowić działanie, mimo, że zdarzenie nie wystąpiło.
- *Mailbox* - „skrzynka pocztowa" używana do synchronicznej wymiany danych pomiędzy procesami. Zawiera dwie kolejki FIFO: procesów wysyłających oraz oczekujących na dane.
- *Tables*, *qtables*, *meters*, *boxes* - używane do zbierania bezpośrednich statystyk związanych ze zdarzeniami (np. czasami pomiędzy przybywającymi zapytaniami). Dotyczą one wartości: maksymalnych, minimalnych, średnich, wariancji, standardowego odchylenia, korelacji, itp.
- *Process class* - używany do segregowania statystyk.
- *Stream of random numbers* - używany do generowania liczb losowych dla ponad dwudziestu różnych rozkładów prawdopodobieństw.

Powyższe elementy mogą być użyte w programie symulacyjnym, umożliwiając użytkownikowi dokładne odwzorowanie struktury i zachowania modelowanych systemów.

Czas jest ważnym pojęciem w modelach symulacyjnych. Czas wykorzystany w symulacji jest całkowicie innym czasem niż czas wykorzystany przez procesor w celu obsługi programu symulacyjnego lub rzeczywisty czas, w którym znajduje się osoba przeprowadzająca symulacje. Czas w CSIM nie ma zdefiniowanej jednostki (np. milisekundy, minuty itp.). Przyjęcie jednostki i konsekwentne jej używanie leży w gestii osoby tworzącej program symulacyjny [126].

Ze względu na dużą popularność pakietu, łatwość konfrontacji otrzymanych wyników badań oraz łatwość publikacji wyników, pakiet ten został wykorzystany i omówiony w dalszej części. Programy symulacyjne utworzono przy wykorzystaniu bibliotek pakietu CSIM w wersji 19 [130].

## 2.4 PRZEGLĄD LITERATURY DOTYCZĄCEJ MODELOWANIA WYDAJNOŚCI

Nowoczesne systemy internetowe stają się coraz bardziej złożone, skalowalne i wydajne. Jednak wydajność architektur systemów internetowych jest funkcją wielu zmiennych. Poprzez powiększanie liczby dostępnych usług i liczby klientów systemy są coraz bardziej „dynamiczne" co powoduje pojawianie się problemów, których przezwyciężenie jest trudne. Architektura tych systemów powinna być elastyczna, aby przyszłe wymagania mogły być łatwo zaspokojone [128]. Nie stworzono dotychczas uniwersalnych architektur, a istniejące sposoby modelowania dotyczą zazwyczaj wybranych elementów. Parametry oraz architektury systemów internetowych w większości przypadków zostały przyjęte a priori przez ich twórców.

Głównym problemem w modelowaniu jest to, że przyrost rozmiaru modelu powoduje, iż przestrzeń stanów rośnie wykładniczo [50]. Większość rozpraw naukowych nie przedstawia weryfikacji otrzymanych wyników z rzeczywistymi systemami internetowymi [48]. Ponadto modele analityczne prezentują jedynie część zagadnień dotyczącą pojedynczych węzłów, ignorując jednocześnie inne komponenty. Autorzy [26] uważają, że rozwój modelowania symulacyjnego ma ogromną wartość praktyczną i jest to przełom dla zastosowań systemów internetowych w biznesie. Dzięki wykrywaniu potencjalnych problemów wydajnościowych podczas budowania systemów, możliwe jest określanie ich przydatności w przyszłości. Oznacza to, że zastosowanie modelowania może polepszyć wydajność usług oferowanych przez systemy internetowe.

Dokonany przegląd literatury dotyczy sposobów modelowania wydajności systemów internetowych. Jego celem jest zaprezentowanie publikowanych modeli wydajnościowych systemów internetowych. Zaprezentowano również artykuły opisujące architektury systemów internetowych.

### 2.4.1 MODELE WYDAJNOŚCIOWE SYSTEMÓW INTERNETOWYCH

Publikowany stan wiedzy na temat modeli wydajnościowych systemów internetowych jest niewielki. Na wstępie przeglądu literatury warto zwrócić uwagę na książkę S. Kouneva [48] obrazującą metody przeprowadzania badań wydajności dla dużych aplikacji rozproszonych systemów internetowych. Proponuje on, aby przeprowadzać analizę wykorzystując następujące modele:

- model biznesowy,
- model funkcjonalny,
- model zachowań użytkowników,
- model zasobów, składający się z podmodeli systemów IT (ang. *Internet Technology*):
    - infrastruktury,
    - kosztów,
    - obciążenia zapytaniami,
    - wydajności.

Istnieje ponadto kilka opracowań skupiających się na różnych sposobach poprawiania czasu reakcji [3, 5, 56, 69, 80, 100]. Szczególnie prace [56, 80, 100] dostarczają modeli

analitycznych pomocnych w skracaniu czasu reakcji systemów internetowych. Nie odnoszą się one jednak do rzeczywistej implementacji.

Artykuły [53, 54] zawierają dyskusję nad problemem przewidywania wydajności i badania przepustowości w rozproszonych systemach WWW. Omówione zostały korzyści oraz problemy wynikające ze stosowania technologii internetowych. Ich autor zastanawia się nad tym, jak złożoność systemów wpływa na poprawną budowę rzeczywistych systemów tak, aby dostarczyć oczekiwany przez klientów poziom usług (QoS) [90]. W dalszej części, na podstawie dostępnej literatury przedmiotu, przedstawiono zagadnienie pomiaru wydajności systemów internetowych dla wybranych modeli.

### MODELE UML DLA SYSTEMÓW INTERNETOWYCH

Modelowanie systemów internetowych przy wykorzystaniu języka UML ogranicza się głównie do ich opisu. W pracy autorzy [55] przedstawiają fazy tworzenia systemów internetowych od momentu analizy. Natomiast autorzy [96] zamodelowali system aukcyjny, który umożliwia adaptację do zmiennych wymagań.

### MODELE KOLEJKOWE DLA SYSTEMÓW INTERNETOWYCH

Sieci kolejkowe są szeroko stosowane do modelowania i analizy złożonych systemów obsługi. Przykłady takich systemów obejmują systemy komunikacji i sieci komputerowych, systemy transakcji, systemy produkcji itp.

Artykuł [50] prezentuje aplikacje J2EE (ang. *Java 2 Platform, Enterprise Edition*) rzeczywistego systemu i pokazuje jak ją modelować w celu poprawy wydajności przetwarzania. Zaprezentowano model sieci kolejkowej do prognozowania wydajności systemu (przepustowości, czasu reakcji i wykorzystania procesorów) dla różnych konfiguracji aplikacji. Wybrane wyniki modelowania zostały potwierdzone poprzez rzeczywiste eksperymenty. Model relatywnie dobrze pozwala na prognozowanie przepustowości i wykorzystania procesorów, a trochę gorzej sprawdza się w przewidywaniu czasu reakcji.

Artykuł [37] definiuje metodologię tworzenia analitycznych modeli sieci kolejkowych dla internetowych systemów e-biznesu. Autorzy, jako jedni z pierwszych, przedstawili eksperymentalną weryfikację czasów odpowiedzi systemów internetowych. Poddają oni w wątpliwość stosowane wcześniej modele analityczne systemów internetowych, które, modelują jedynie wybrane komponenty.

Van der Mei i inni [70] zamodelowali serwer WWW jako pojedynczy system kolejkowy. Swojego modelu użyli do przewidywania charakterystyk jego wydajności.

Beckers i inni [16] opracowali model wydajności bazujący na serwerach WWW. Zaznaczyli, że modele wydajności systemów klient-serwer muszą uwzględniać interakcje różnych procesów, biorących udział w przetwarzaniu zapytań.

Autorzy [73] używają modeli kolejkowych oraz techniki MVA (ang. *Mean Value Analisis*) [33, 65] do modelowania systemów internetowych klient-serwer. Ci sami autorzy w pracy [72] prezentują model serwera w postaci zamkniętej sieci kolejkowej. W sieci tej serwer WWW otrzymuje zapytania od ustalonej liczby klientów. Zbiór klientów jest reprezentowany w sieci kolejkowej jako zasób opóźniający, ponieważ obsłużone przez serwer zapytania po przejściu przez sieć trafiają bezpośrednio do klienta bez oczekiwania w kolejce. Czas obsługi zapytania na stanowisku obsługi równy jest średniemu czasowi spędzonemu przez klienta na wygenerowaniu kolejnego zapytania. Sieć reprezentowana jest przez zasób zależny od obciążenia. Dla wzrastającego obciążenia maleje przepustowość sieci ze względu na wzrastającą liczbę kolizji przy próbie dostępu do medium transmisyjnego.

W pracy [22] tworzone są proste modele kolejkowe w postaci systemu kolejkowego FIFO dla serwera WWW. Autorzy [22] prezentują również model kolejkowy *M/G/1/PS* dla serwera WWW. Prosty model odwzorowuje mniejszą liczbę parametrów, które są prostsze do oszacowania.

Modele kolejkowe są często wykorzystywane do opisu systemów sieciowych. Cytowane wcześniej rozwiązania, szczególnie Kouneva [50], są ciekawym i nowatorskim spojrzeniem na problematykę modelowania warstwowych systemów internetowych. Ze względu na złożoność problemu modelowania systemów, większość publikacji ogranicza się do modelowania wybranych elementów najczęściej przy użyciu pojedynczych systemów kolejkowych. Do symulacji sieci kolejkowych opracowano wiele narzędzi [150] np.: PEPSY-QNS [148], RAQS [46, 155] czy QNAT [68, 88, 108].

### MODELE SIECI PETRIEGO DLA SYSTEMÓW INTERNETOWYCH

W literaturze można znaleźć szereg modeli bazujących na sieciach Petriego. Sieci Petriego, a szczególnie kolorowane sieci Petriego czy kolejkowe sieci Petriego, używane są do modelowania systemów internetowych.

Wells i inni [115] wykonali analizę serwerów WWW używając kolorowanych sieci Petriego. Wartości parametrów modelu pozyskiwane są dzięki eksperymentom przeprowadzonym na systemach rzeczywistych. W artykule przedstawiono ogólną strukturę modelowania i analizy rozproszonego środowiska przetwarzania. Opracowany model w języku kolorowanych sieci Petriego dotyczy głównie opisu systemu. W kolejnym artykule [30] wspomniani autorzy zaproponowali podział modelu na trzy warstwy (konstrukcyjną, aplikacji, zasobów) przedstawione jako „podstrony", gdzie każda z nich modeluje inną część systemu. Stosowanym przez nich narzędziem jest np. CPN Tools [134].

Kounev i Buchmann w artykule [51] przedstawiają modele QPN systemu internetowego. Modele QPN mogą zostać wykorzystane do analizy wydajności. Autorzy zademonstrowali sposób wykorzystania formalizmu QPN. Pokazują, że QPN jest użytecznym narzędziem przewidywania np. czasu odpowiedzi. Do prowadzenia symulacji wykorzystali oni narzędzie QPN-Tool [9, 10, 11]. W pracy [49] Kounev wprowadza założenie, że połączenie z każdym serwerem podczas przetwarzania operacji nawiązywane jest tylko raz i nie ma możliwości ponownego powrotu w to samo miejsce modelu.

W artykule [81] autorzy zaprezentowali ogólny model systemów bazujący na SPN, opisujący zachowanie systemów równoległych w następstwie wystąpienia błędów. Zastosowane podejście pozwoliło na określenie parametrów niezawodności modelu.

Zaprezentowane modele bazujące na sieciach Petriego są kolejnym skutecznym mechanizmem modelowania systemów internetowych. Proponowane modele systemów, a szczególnie modele CPN i QPN, mogą być stosowane do analizy wydajnościowej. Podane przykłady ograniczają się jednak do oddzielnego modelowania poszczególnych elementów warstw systemów i założenia jednokierunkowości modelu oraz pośredniego połączenia modelu z rzeczywistymi systemami.

### 2.4.2 MODELE ARCHITEKTUR SYSTEMÓW INTERNETOWYCH

Treścią podrozdziału są różne modele odnoszące się bezpośrednio do architektur systemów internetowych [16, 27]. Model architektury można podzielić na odpowiednie warstwy i zadania jakie one wykonują:

- na początku interakcji z systemem, przeglądarka WWW wysyła żądanie przesłania strony,
- internetowy serwer prezentacji wysyła stronę do klienta lub komunikuje się, w razie konieczności, z serwerem aplikacji,
- serwer aplikacji zgodnie z otrzymanymi instrukcjami wysyła zapytanie do bazy danych, która jest centralnym punktem systemu,
- transakcja zostaje wykonane na przechowywanych danych i zwrócony zostaje zbiór rekordów wynikowych,
- serwer aplikacji po otrzymaniu danych umieszcza je w odpowiednich miejscach na stronie, którą ogląda użytkownik poprzez przeglądarkę internetową.

Ponieważ większość systemów internetowych używa zaprezentowanej, wielowarstwowej architektury, wiele publikacji skupia się na problemie analitycznego modelowania ich zachowań. Dla przykładu, autorzy [111] zaprezentowali model bazujący na sieci kolejek, gdzie kolejki reprezentują różne warstwy systemu. Eksperymenty pokazały użyteczność modelu dla wykrywania „wąskich gardeł" (ang. *bottleneck*). Potwierdzono skuteczne działanie modelu wykorzystując dwie darmowe aplikacje, uruchomione na grupie serwerów linux'owych.

Kolejnym sposobem poprawy wydajności systemów poprzez zmianę architektury jest klastrowanie [21]. W związku ze zwiększającym się ruchem w sieci, systemy bazujące na klastrach stają się coraz ważniejsze. Głównym problemem jest wykorzystanie dostępnych zasobów klastrowych. Klaster może składać się z wielu węzłów, dlatego potrzebne są efektywne mechanizmy decydujące, który z węzłów powinien obsłużyć poszczególne zapytania, by w pełni wykorzystać dostępne zasoby. Jednym z podejść do tego problemu jest użycie rozdzielania obciążania w postaci rozdzielacza/dyspozytora (ang. *dispatcher*). Jest on usytuowany przed węzłami przetwarzającymi i spełnia rolę wejścia do systemów. Wszystkie decyzje o rozdzieleniu zapytań podejmowane są przez dyspozytora [58, 59, 63]. W klastrze dyspozytor rozsyła nadchodzące zapytania [4]. W opracowaniu [99] autorzy przedstawiają serwer bazujący na klastrze, który potrafi zoptymalizować przepustowość i znacznie polepszyć osiągi względem dotychczasowych rozwiązań. Kolejnym rozwiązaniem jest LVS (ang. *Linux Virtual Server*) [143]. Jest to struktura, której ruch jest dzielony na fizyczne serwery. Stosowane są również metody przydzielające ruch do serwerów na podstawie ustalonych wag lub po analizie aktualnego stanu ruchu lub obciążenia. Dodatkowo w celu wydajnego trasowania zapytań do serwerów rozwinęły się różne techniki wyrównywania obciążeń [43]. Strony internetowe mogą znajdować się w wielu lokalizacjach. Podstawową techniką dla zwiększania wydajności jest buforowanie. Kluczowym problemem buforowania jest utrzymywanie aktualności bufora. Trudność polega na tym, iż nie jest możliwe przewidzenie, kiedy dane stracą ważność. W artykule [47] zaproponowano kolejne rozwiązanie rozkładania ruchu, podobne do LVS, które poprzez zastosowanie szeregowania bazującego na sprawdzaniu wykorzystania pamięci, wybiera węzeł do przetwarzania zapytań. Z kolei propozycja architektury przedstawiona w [109] polega na różnicowaniu zapytań przychodzących do serwera WWW. Autorzy zaimplementowali algorytm adaptacyjny, który dopasowuje poziom priorytetu i określa, jak i z jaką wagą dane zapytanie będzie traktowane. Artykuł [98] przedstawia prosty,

lecz efektywny mechanizm kontroli przeciążeń dla serwerów sieciowych. Walidacja proponowanego mechanizmu została zweryfikowana poprzez zaimplementowanie go dla rzeczywistego serwera sieciowego oraz poddanie testom wydajności. Artykuł [38] prezentuje metodę dostarczenia kontroli dostępu oraz rozdzielania zapytań. Autorzy, poprzez wprowadzenie zewnętrznego mechanizmu wyceny poszczególnych zapytań oraz wprowadzenie swoistego mechanizmu określania kosztów ich przetwarzania, doprowadzili do wyodrębnienia różnych typów zapytań, osiągając stabilne zachowanie systemu i skrócenie czasów odpowiedzi. Inne tego typu propozycje wymagają modyfikacji w systemach operacyjnych lub całkowitej modyfikacji serwerów. Proponowana metoda nie wymaga żadnych zmian w kodzie źródłowym, oprogramowaniu czy bazie danych. Podsumowaniem sposobów rozdzielania ruchu może być artykuł [23], który prezentuje przegląd architektur rozproszonych systemów internetowych.

Kolejnym istotnym elementem systemów internetowych jest baza danych. Modelowanie jej zachowania jest niezwykle trudne, ze względu na mnogość czynników wpływających na jej działanie, szczególnie w przypadku zastosowania replikacji. Modyfikacja architektury w przypadku baz danych polega na zastosowaniu mechanizmu replikacji w celu zwiększenia dostępności danych, równomiernego rozłożenia obciążenia serwerów oraz uzyskania niezawodności i skalowalności. Sama replikacja baz danych w węzłach może spowodować problemy spójności, które muszą być kontrolowane. Podstawową własnością systemów internetowych jest to, aby części *front-end* i *back-end* były zawsze spójne. W artykule [124] zaproponowano algorytm replikacji, który zapewnia tę własność. Użyto leniwej replikacji (ang. *lazy replication*), gdzie każda zmiana w bazie zostaje wysłana do replik pod koniec wykonania transakcji. Zastosowany protokół zgodności gwarantuje spójność między stanem wszystkich replik a główną bazą danych. Artykuł [77] prezentuje rozwiązanie replikacji danych, które zapewnią dużą spójność bez ograniczeń replikacji gorliwej (ang. *eager replication*). Węzły mogą zawierać niezależne, niejednorodne bazy danych, które są rozważane jako „czarne skrzynki". Zaprezentowano replikację *multi-master* [83] i pokazano wszystkie komponenty potrzebne do implementacji. Rozwiązanie przetestowano na prototypie klastra ośmiowęzłowego i zaprezentowano wyniki eksperymentu, które pokazują, że algorytm wprowadza, pomijalną według autorów, utratę aktualności danych. Ciekawym rozwiązaniem w dziedzinie modelowania wydajności bazy danych są artykuły [119, 121] oraz praca [120]. Autor modeluje i bada różne typy replikacji bazy danych z wykorzystaniem symulatora CSIM [130] w celu określenia ich wydajności. W przypadku replikacji nie napotkano formalnych modeli obrazujących wydajność.

## 2.5 PODSUMOWANIE ROZDZIAŁU

Modeli systemów internetowych jest wiele jednak brakuje opracowań dla ISIROSO. Dostępne w publikacjach naukowych rozwiązania w większości przypadków dotyczą jedynie niektórych elementów systemów z przetwarzaniem jednokierunkowym. Rozwiązania te nie są idealne ze względu na oddziaływanie poszczególnych elementów systemów na siebie, powodujące powstawanie zależności, których w większości przypadków nie wolno zaniedbać. Dodatkowo, brak jest w źródłach publikowanych informacji na temat modelowania systemów szybkozmiennych w kontekście wydajności, w połączeniu z proponowanymi elementami architektury. W wyniku przeglądu literatury wykazano, że mimo istnienia modeli systemów internetowych, brak jest modeli wydajnościowych systemów szybkozmiennych.

Modelowanie nie jest procesem prostym, dlatego też w celu osiągnięcia zadawalających wyników należy stosować różne - najlepiej - uzupełniające się i weryfikujące nawzajem modele. Spośród zaprezentowanych rozwiązań wybrane zostały formalne: modele czasowych kolorowanych sieci Petriego i nieformalne: modele w języku UML oraz CSIM. Dokonano wyboru opisu w języku UML ze względu na przydatność do zaprezentowania ISIROSO i przedstawienia szybkozmienności ofert systemów internetowych. Modele sieci kolejkowych, użyte do opisu systemów klasy ISIROSO, utworzono na podstawie istniejącej wiedzy w tym temacie oraz własnych doświadczeń. Ogólny model węzła bazuje na modelu przedstawionym w [14, 18]. Na bazie tych modeli utworzono modele wydajnościowe TCPN i CSIM. Modele czasowych kolorowanych sieci Petriego składają się z, skonstruowanych w tym celu, systemów kolejkowych FIFO i PS. Zastosowanie nieformalnej metody modelowania, jaką jest CSIM ma na celu sprawdzenie prowadzonych badań na formalnych modelach TCPN. Wymienione metody i narzędzia są podstawą modelowania ze względu na możliwość modelowania parametrów wydajnościowych oraz brak podobnych zastosowań w opublikowanej literaturze przedmiotu rozprawy. Szczegółowy opis budowy modeli zaprezentowano w rozdziałach 3 i 4, a ich analizę w rozdziale 5.

Równie często prezentowane są w literaturze propozycje eksperymentalnego podejścia do budowy systemów. W opracowaniu [67] autorzy przedstawiają optymalizację eksperymentalną, polegającą na modyfikacji parametrów rzeczywistego serwera warstwy prezentacji dla zmieniających się obciążeń. Monitorując użycie zasobów, zmieniając jednocześnie parametry serwera, autorzy minimalizują czas jego reakcji. Autorzy [75] zbudowali działający internetowy system zakładów wyścigów konnych przetwarzający zapytania w czasie rzeczywistym. Na podstawie danych, otrzymywanych w wyniku monitorowania działania systemu, modyfikują jego architekturę. W celu poprawy działania stosuje się coraz to nowsze technologie oraz szybsze urządzenia. Proponowane podejście nazwać można całkowicie eksperymentalnym. Jest kosztowne, czasochłonne oraz pociąga za sobą konieczność modyfikacji w czasie eksploatacji, ale jest często stosowane w praktyce. Zaprezentowane rozwiązania eksperymentalne są użyteczne, ale nie posiadają żadnych formalnych podstaw. Wydaje się, że dopiero równoczesne zastosowanie modelowania, dającego możliwość formalnego opisu i eksperymentów pozwalających na weryfikację konstruowanych modeli, daje dużo bardziej użyteczne rezultaty.

# 3 MODELE ISIROSO

Celem rozdziału jest wskazanie klasy systemów internetowych z szybkozmiennymi ofertami, dla której możliwe będzie opracowanie modeli wydajnościowych przy wykorzystaniu metod formalnych i nieformalnych. Systemy te są trudne do zrównoleglenia, gdyż ze względu na sekwencyjność obsługi, realizacja transakcji zależy od transakcji poprzednich. Zatem dążenie do zrównoważenia ISIROSO jest trudne, jednak wykonalne w przypadku rozróżnienia zleceń niepowiązanych ze sobą.

W rozdziale przedstawiono propozycje różnych modeli dla ISIROSO. Modele te będą się także odnosić do giełdy internetowej jako przedstawiciela tej klasy systemów. Do prezentacji systemu giełdy internetowej zastosowano wybrane diagramy języka UML. Zaproponowano narzędzia programowe, wspomagające modelowanie i analizę wydajnościową środowisk internetowych z zastosowaniem czasowych kolorowanych sieci Petriego (TCPN) oraz bibliotek oprogramowania symulacyjnego CSIM [130]. Tak utworzone modele ISIROSO są analizowane przy wykorzystaniu:

- sieci TCPN, z zastosowaniem modułu analizy wydajnościowej pakietu symulatora Design/CPN [35],
- aplikacji symulującej, napisanej z wykorzystaniem bibliotek pakietu symulatora wydajności CSIM [149].

## 3.1 E-BIZNES

Biznes elektroniczny (e-biznes (ang. *e-business*)), jest działalnością gospodarczą, obejmującą transakcje biznesowe, której zyski pochodzą w całości lub częściowo z Internetu. Pojęcie to obejmuje między innymi wymianę informacji między producentami, dystrybutorami oraz odbiorcami produktów i usług, zawieranie kontraktów, przesyłanie dokumentów, prowadzenie telekonferencji, pozyskiwanie nowych kontaktów, wyszukiwanie informacji, kupno lub sprzedaż towarów, itp. E-biznes obejmuje wszelkie przejawy komercyjnego wykorzystania Internetu.

Handel elektroniczny (e-handel, e-commerce) jest specyficznym typem e-biznesu związanym z prowadzeniem nastawionych na zysk transakcji biznesowych przez sieć globalną. Systemy internetowego e-handlu polegające na elektronicznej sprzedaży towarów i usług z wykorzystaniem technologii oferowanych przez Internet, są coraz bardziej popularne. Handel elektroniczny może być [158]:

- bezpośredni (ang. *direct electronic commerce*) oznacza zamawianie i otrzymywanie towarów przez sieć globalną,
- pośredni (ang. *indirect electronic commerce*) oznacza zamawianie towarów przez siec internetową a otrzymywanie ich metodami tradycyjnymi (np. pocztą).

E-handel dzieli się, podobnie jak rynek tradycyjny, na modele:

- brokerski - na którym sprzedawca oferuje swój towar a klient określa czy chce go zakupić,
- dopasowany do indywidualnych potrzeb klienta - charakteryzujący się tym, że klient określa zapotrzebowanie na jakiś produkt a sprzedawca zajmuje się jego dostarczeniem,

- kontaktowy - polegający na pośredniczeniu między sprzedawcą a klientem.

Każdy rynek jest, z reguły, zdominowany przez określoną część klientów. Ze względu na tę dominację rynek można sklasyfikować na dwa sposoby:

- pierwszy sposób bierze pod uwagę napędzającą go „siłę", a wyróżnia się w nim trzy grupy:
  - rynek sprzedawcy - na którym występuje niewielka grupa sprzedających oraz duża liczba kupujących,
  - rynek nabywcy - gdzie spotyka się duża grupa dostawców z niewielką grupą odbiorców,
  - rynek otwarty - charakteryzujący się podobną grupą nabywców i sprzedawców,
- drugi sposób opisuje rynek ze względu na liderów i wyłania także trzy grupy:
  - rynek ofertowy - gdzie sprzedawcy prezentują własne produkty kupującym,
  - rynek popytowy - na którym klienci przedstawiają specyfikacje produktów, a dostawcy tworzą odpowiednio dopasowane oferty do prezentowanej specyfikacji,
  - rynek otwarty - będące odpowiednikiem rynku otwartego określanego z perspektywy „siły" napędowej.

Handel elektroniczny, ze względu na strony transakcji, można podzielić na [158]:

- B2B (ang. *Business to Business*) - handel pomiędzy firmami i instytucjami wykorzystującymi Internet,
- B2C (ang. *Business to Customer*) - zawieranie transakcji z indywidualnymi klientami poprzez Internet,
- C2C (ang. *Customer to Customer*) - handel pomiędzy indywidualnymi klientami,
- B2P (ang. *Business to Public*) - marketing elektroniczny nie związany bezpośrednio ze sprzedażą towarów i usług.

E-handel ze względu na rodzaj transakcji dzieli się między innymi na [89]:

- aukcje (dzieł sztuki, towarów, usług),
- biblioteki elektroniczne,
- dystrybucje towarów (systemy zarządzania towarami (np. spedycja)),
- e-learning (szkolenia on-line),
- giełdy (np. papierów wartościowych, walutowe),
- internet banking (bankowy systemy internetowy),
- newsletter (przesyłanie informacji na skrzynkę emailową),
- przeglądarki i wyszukiwarki informacji,
- rezerwacje biletów (np. lotniczych, do kina, na koncert, na mecz),
- sklepy,
- systemy lokalizacji pojazdów,
- wywoływanie zdjęć on-line,
- zakłady sportowe.

Giełda internetowa odniesiona do poprzednich definicji to bezpośredni, brokerski rynek otwarty transakcji B2C e-handlu. Giełdy dzieli się na:

- giełdy papierów wartościowych (np. [132, 145, 151]),
- giełdy walutowe (np. [137, 157, 159, 160, 164]).

Istnieją systemy internetowe, oferujące możliwość „gry" na giełdzie, głównie walutowej, w „czasie rzeczywistym". Realizują one dostęp do konta inwestycyjnego ze stron WWW lub

poprzez zainstalowane oprogramowanie dostępowe na komputerze gracza (por. dodatek). W Polsce brak jest internetowego rozwiązania dla giełdy papierów wartościowych [132, 145, 151] mimo tego, że jedna czwarta rachunków inwestycyjnych to rachunki internetowe [132]. Skutkuje to koniecznością korzystania z pośredników - biur maklerskich [152] - co jednoznacznie wiąże się z dodatkowymi opóźnieniami, na które klient nie ma wpływu. Rozwiązanie takie, jak powszechnie wiadomo, nie jest najlepsze ze względu na pojawiające się opóźnienia.

Jednym z głównych zadań stawianych ISIROSO jest dostarczanie aktualnych danych w ustalonych ramach czasowych [118]. Jest to utrudnione ze względu na dużą liczbę potencjalnych klientów internetowych (Polska - kilka milionów, świat - kilkaset milionów [144]) mogących korzystać jednocześnie z systemu. Z grubsza można wyróżnić następujące grupy klientów:

- zwykłych - wysyłających do około kilkunastu [zapytań/s],
- obciążających - generujących około kilkuset [zapytań/s],
- szczytowo obciążających - przekazujących powyżej kilkunastu tysięcy [zapytań/s] [138, 145].

Do systemów szczytowo obciążonych należą giełdy internetowe [145].

## 3.2 MODELE UPROSZCZONEJ GIEŁDY INTERNETOWEJ

Rzeczywista giełda składa się z następujących systemów notowań:

- ciągłych - polegającego na ciągłej zmianie ceny,
- jednolitych z dwoma fixingami - czyli z dwukrotnym określeniem kursu.

W dalszych rozważaniach ograniczono się do systemu notowań ciągłych ze względu na ciągłą możliwość zmiany cen akcji (por. dodatek). W takim systemie transakcje są zawierane po kursie najlepszego oczekującego zlecenia. Do określenia kursu przyjęto, podobnie jak w rzeczywistej giełdzie, maksymalizację obrotu, czyli sytuacją w której występuje największa liczba kupujących i sprzedających jednocześnie.

Uproszczony model giełdy, jako systemu notowań papierów wartościowych, bazuje na obrocie najbardziej znanymi papierami wartościowymi - akcjami[2]. W użytym dalej uproszczonym modelu wprowadzono:

- ograniczenie wielkości zlecenia,
- sprawdzanie liczby dostępnych akcji,
- sprawdzanie dostępności środków płatniczych,
- sprawdzanie liczby posiadanych akcji.

Pozostawione zostały dwie główne zasady pierwszeństwa giełdy [132]:

- cena - realizacja zleceń o najwyższej cenie,
- czas - w przypadku identycznej kwoty transakcji realizacja tej, która była w systemie pierwsza.

Interesującym przykładem giełdy papierów wartościowych jest Warszawska Giełda Papierów Wartościowych (WGPW) [132], dlatego wybrana ona została jako wzorzec budowanego systemu. WGPW operuje na wielu instrumentach finansowych takich, jak akcje, obligacje, prawa poboru, certyfikaty inwestycyjne, kontrakty terminowe itd. Różnorodność

[2] Akcje są to papiery wartościowe stwierdzające uczestnictwo właściciela w kapitale spółki akcyjnej powiązane bezpośrednio z prawem do dywidendy czyli partycypacji w zyskach spółki.

instrumentów finansowych pociąga za sobą skomplikowany mechanizm systemu notowań obsługującego inwestorów. Ze względu na swoje skomplikowanie, system obsługujący giełdę jest scentralizowany. Nie pozwala to na bezpośrednie składanie zleceń przez Internet. Inwestorzy muszą korzystać z rachunków biur maklerskich jako pośredników pomiędzy nimi a giełdą.

### 3.2.1 MODEL FORMALNY

W podrozdziale przedstawiono ogólny opis klasy systemów określonej jako ISIROSO [85, 86, 89].

**Definicja 3.1.**
Systemy Interaktywne (SI) są to systemy pozwalające na bezpośrednią (on-line) wymianę informacji, pomiędzy klientem a systemem, odbierając zapytania i reagując na nie.

**Definicja 3.2.**
Interaktywne Systemy Internetowe (ISI) to systemy interaktywne (SI) (Def. 3.1) realizujące zapytania internetowe.

**Definicja 3.3.**
Giełdowe ISI to struktury pozwalająca na dokonywanie obrotu towarem, np. papierami wartościowymi, określająca dostęp do informacji (np. o stanie indeksów giełdowych[3]), po stronie których znajduje się kompletna oferta zawierająca informacje o przedmiocie transakcji - towarze, cenie i ilości.

**Definicja 3.4.**
Ofertą systemów $O$ nazywa się trójkę $O = (F, I, C)$, gdzie:

- $F$ - oznacza rodzaj oferowanego towaru,
- $I$ - oznacza ilość towaru będącego przedmiotem oferty,
- $C$ - oznacza cenę.

Po stronie klienta znajdują się dwa typy zleceń: kupna i sprzedaży. Przyjęto konwencję, że zlecenia dla $I > 0$ nazywa się zleceniami kupna, zaś zlecenia, gdzie $I < 0$ zleceniami sprzedaży.

**Definicja 3.5.**
Niech $B = \{b_k, k = 1, \ldots, N\}$ będzie bazą danych (zbiorem) ofert kupna i sprzedaży, gdzie: $N$ oznacza rozmiar, zaś $b_k$ - $k$-ty element bazy danych.

[3] Indeks giełdowy to wartość obliczana na podstawie wyceny akcji spółek określająca koniunkturę na giełdzie.

W celu ujednolicenia prowadzonych dalej rozważań wprowadzono dwa zbiory:

- chwil czasowych $ZCC \in R_+$,
- odcinków czasu $ZOC \in R_+$.

**Definicja 3.6.**

Stanem $S_B$ w chwili $t_{S_B} \in ZCC$ bazy danych ofert ($B$) nazywa się zbiór danych o rodzajach towarów, mogących być przedmiotem zlecania kupna/sprzedaży i cenie. W dalszej części przez stan bazy danych ($S_B$) rozumie się stan bazy w chwili przesłania zapytania klienta do systemów.

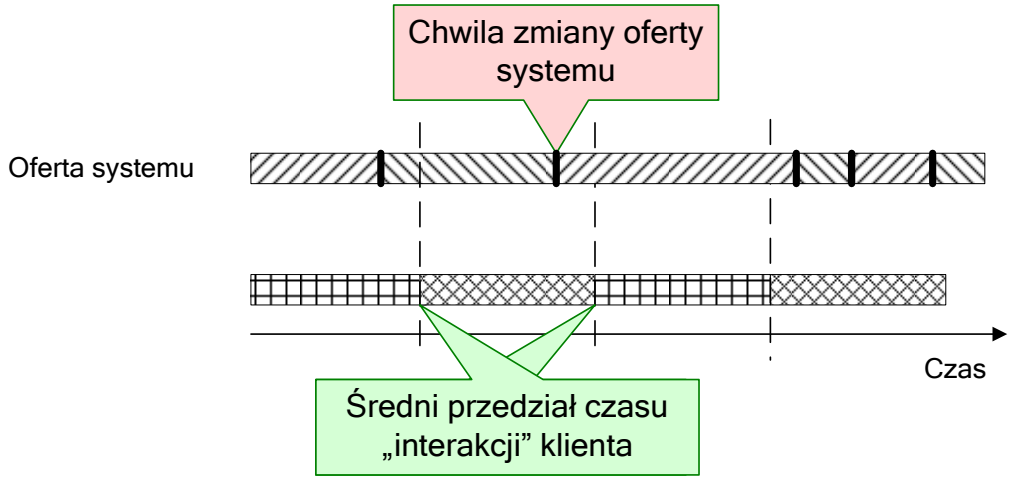
**Definicja 3.7.**

Poprzez transakcję $Tr$ zapytania klienta rozumie się szereg operacji dokonywanych na bazie danych, w konsekwencji których może zmienić się stan bazy danych $S_B$.

Przez $S_{B+1}$ w chwili $t_{S_{B+1}} \in ZCC$ oznacza się kolejny stan bazy danych spowodowany modyfikacją oferty systemu $O$ przez transakcję $Tr$. W przypadku, gdy ilość kupowanego/sprzedawanego towaru jest większa od określonych progów granicznych, oferta systemów $O$, poprzez zmianę ceny, ulega modyfikacji i kolejny stan bazy danych różni się od poprzedniego $S_{B+1} \neq S_B$. Natomiast w przypadku, gdy np. transakcja nie została zrealizowana lub spowodowana przez nią modyfikacja zawiera się w określonych granicach ilości kupowanego/sprzedawanego towaru, stan bazy danych pozostaje niezmieniony i $S_{B+1} = S_B$.

SZYBKOZMIENNOŚĆ SYSTEMÓW

Wśród systemów interaktywnych można wyróżnić tzw. systemy szybkozmienne tj. takie, w których oferty zmieniają się często, co powoduje odrzucanie części zapytań od klientów. Czas zmiany oferty systemów jest porównywalny z czasem oczekiwania na realizację transakcji opowiadającej zapytaniu klienta lub z czasem przygotowania i złożenia zapytania - tzw. przedziałem czasu „interakcji” klienta (Rys. 3.1).




Rys. 3.1 Reprezentacja graficzna szybkozmienności przykładowej oferty systemu

Dalsze rozważania i przykłady będą dotyczyć internetowego systemu giełdowego. Przedstawiona giełda zawiera informacje o nazwach spółek, liczbie oraz cenie za akcje. Formalnie giełdę można opisać w następujący sposób.

**Definicja 3.8.**
Niech $ZPW$ oznacza zbiór papierów wartościowych na giełdzie (rodzaje towarów z definicji 3.4), $I$ - liczba akcji (ilość towaru) zaś $ZC$ - zbiór cen. Ofertą systemu giełdowego nazywa się trójkę:

$$O_G = (ZPW, I, ZC) \tag{3.1}$$

W takim ujęciu giełda jest pewną bazą danych (zbiorem) ofert ($B$). Jej oferta $O_G$ zmienia się w czasie co powoduje zmianę stanu bazy danych $S_B$ a tym samym zmianę oferty systemu giełdowego. Zmiany w ofercie systemu dotyczą dwóch szczególnych przypadków:

- zdarzeniowego - polegającego na ciągłej modyfikacji cen akcji spółek w trakcie trwania ciągłej sesji giełdowej, uzależnionej zarówno od kupna, jak i sprzedaży,
- czasowego - polegającego na modyfikowaniu cen po zakończeniu okresu czasu trwania sesji giełdowej.

Wprowadzono pojęcie giełdy internetowej.

**Definicja 3.9.**
Giełdą internetową nazwa się ciąg sesji:

$$CS = \{D_z\}_{z=1}^{\infty} \tag{3.2}$$

gdzie: $D_z$ jest *z*-tą sesją.

Przyjmuje się, że czas trwania każdej sesji $t_{D_z}$ jest stały:

$$t_{D_z} = \text{const} \tag{3.3}$$

Przez $ZK$ oznaczono zbiór klientów/inwestorów [117]. Zakłada się, że każdy z inwestorów $K \in ZK$ może składać zapytania internetowe - „grać" na giełdzie. Uwzględniając to, że dane zapytanie wysyłane jest przez konkretnego inwestora. Zapytanie klienta oznaczono przez $Z_K = (ZK, ZS, I, ZC)$, gdzie $K \in ZK$, a pozostałe parametry zgodne są z definicją 3.8. Zlecenie $Z_K$ oznacza kupno/sprzedaż - zgodnie z definicją 3.4 - przez klienta $K$ akcji spółek w ilości $I$ i po cenie $C \in ZC$. W przypadku zlecenia kupna $I > 0$, cena $C$ oznacza de facto maksymalną cenę po jakiej dany klient zobowiązuje się kupić akcje spółki będące przedmiotem oferty $O_G$. Jeśli $Z_K$ jest zleceniem sprzedaży to $I < 0$, to $C$ oznacza minimalną cenę po jakiej klient chce sprzedać akcje.

Poniżej przedstawiony został model formalny realizowanych zleceń w postaci transakcji na bazie danych.

Niech $ZTr$ będzie zbiorem transakcji na bazie danych:

$$ZTr = \{Tr_1, ..., Tr_l\} \tag{3.4}$$

gdzie:

- $Tr_i$ - oznacza *i*-tą transakcję $Tr$,
- $L$ - oznacza liczbę transakcji w czasie sesji.

Transakcja, należąca do $ZTr$, jest to zbiór operacji $ZOp$ wykonywanych na bazie danych (Wzór 3.5). Zbiór operacji transakcji $Tr_i$ można zapisać jako:

$$ZOp_{Tr_i} = \{Op_i^1, ..., Op_i^j\} \tag{3.5}$$

gdzie $j$ jest liczbą operacji $i$-tej transakcji. Wykonywanie transakcji $i$-tej zawsze utożsamia się konkretną ofertą $O_G$.

Jeśli $t_{Tr_i} \in ZOC$ oznacza czas trwania $i$-tej transakcji, zaś $t_{Op_i^j} \in ZOC$ czas trwania $j$-tej operacji $i$-tej transakcji, to musi zachodzić warunek:

$$\sum_{j=1}^{j} t_{Op_i^j} = t_{Tr_i} \tag{3.6}$$

Działania wchodzące w skład *i*-tej operacji transakcji $Tr_i$ to zbiór:

$$Op_i^j = \{czytaj(b_k)_i, sprawdz(b_k)_i, zapisz(b_k)_i, przerwij\_Tr_i, zakoncz\_Tr_i\} \tag{3.7}$$

gdzie:

- $czytaj(b_k)$ - to operacja czytania danych,
- $sprawdz(b_k)$ - oznacza operacje sprawdzenia możliwości realizacji transakcji poprzez weryfikację aktualności oferty [117],
- $zapisz(b_k)$ - to operacja zapisywania danych,
- $przerwij\_Tr$ - oznacza operację przerwania,
- $zakoncz\_Tr$ - oznacza pomyślne zakończenie realizacji transakcji.

Wyróżnić należy operację $sprawdz(b_k)$, aby podkreślić ważność konieczności sprawdzania aktualności oferty systemu.

**Definicja 3.10.**

Niech $tz_{Tr_i} \in ZCC$ będzie czasem złożenia *i*-tego zapytania, dowolnego *K* klienta ze zbioru inwestorów ($ZK$). System internetowy należy do klasy ISIROSO wtedy, gdy zmieniający się stan bazy danych $S_{B+1}$ różni się od poprzedniego stanu $S_B$. W tym przypadku średni czas jaki upływa od zmiany oferty, będącej jednocześnie zmianą stanu bazy danych, jest porównywalny lub krótszy od średniego czasu interakcji użytkownika, czyli wysłania kolejnego zapytania $tz_{Tr_{i+1}} \in ZCC$ - zgodnie z (Rys. 3.1):

$$t_{S_{B+1}} - t_{S_B} \leq tz_{Tr_{i+1}} - tz_{Tr_i} \tag{3.8}$$

gdzie:

- $t_{S_B}$ - jest czasem wystąpienia aktualnego stanu bazy danych i $t_{S_B} \in ZCC$,
- $t_{S_{B+1}}$ - jest czasem wystąpienia kolejnego stanu bazy danych ze zmienioną ofertą systemu i $t_{S_{B+1}} \in ZCC$.

Cechy ISIROSO to: sekwencyjność obsługi zapytań, istnienie ograniczenia czasu na realizację zapytania, uwarunkowanie od zachowania innych klientów w tym samym czasie oraz aktualność oferty.

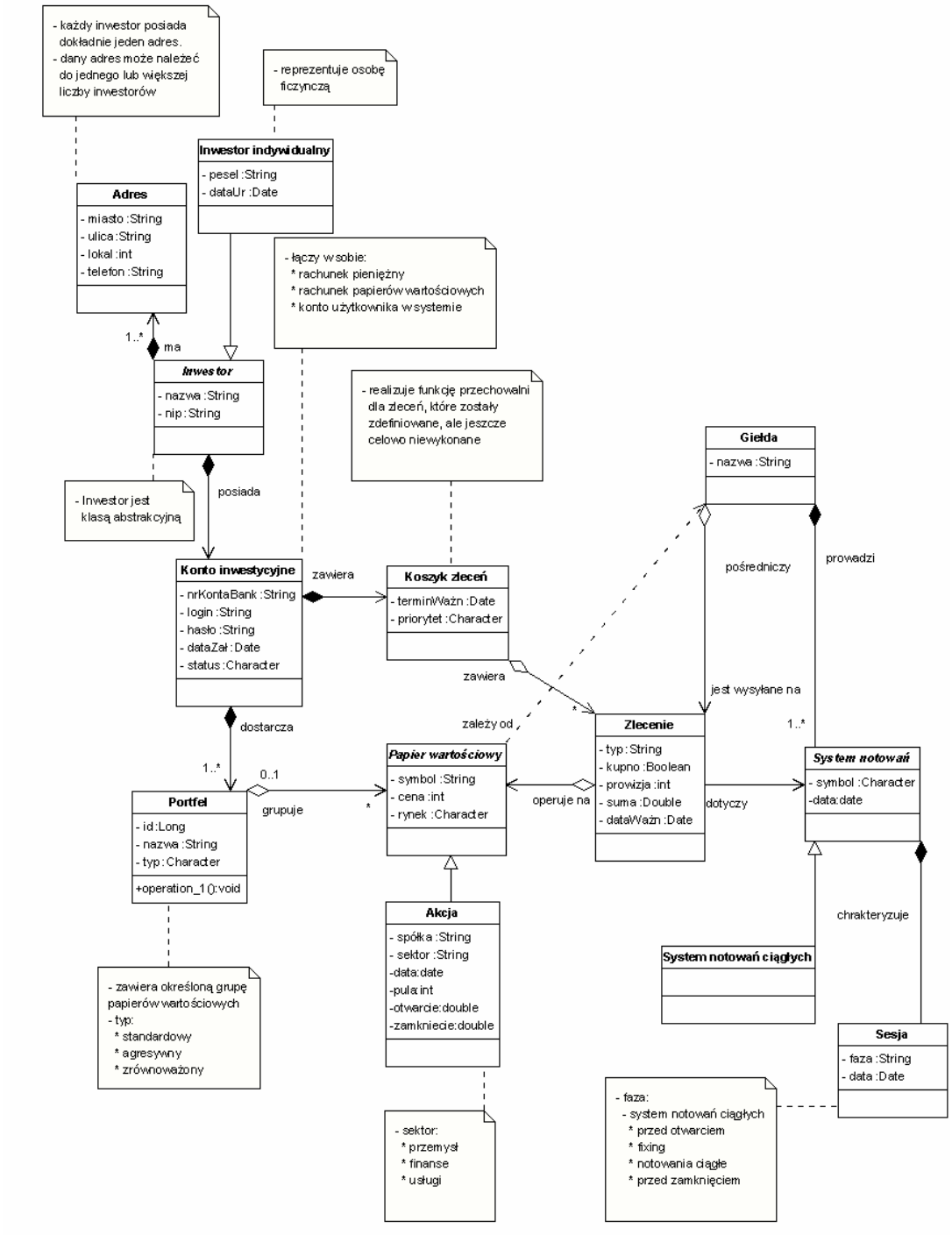
Prezentowany system internetowy w postaci giełdy internetowej można więc zakwalifikować do grupy ISIROSO, dla których zasadniczą rolę, w realizacji zapytania, odgrywa czas napływania zleceń i czas modyfikacji oferty systemu. Do szybkozmiennych systemów internetowych zaliczyć można również systemy aukcyjne czy zakłady sportowe.

Realizacji zapytania wysłanego w transakcji $Tr_i$ dowolnego klienta $K \in ZK$ nie zawsze jest możliwa. Ten niekorzystny stan potęgowany jest szczególnie w przypadku systemów obciążonych zapytaniami. W przypadku ISIROSO duża część zapytań spełnienia warunek (Wzór 3.8). W tej sytuacji klienci ( $ZK$ ) zmuszeni są do oczekiwania na realizację swoich zapytań. Czas odpowiedzi systemu wydłuża się co doprowadza do jego niezrównoważenia.

### 3.2.2 MODEL W JĘZYKU UML

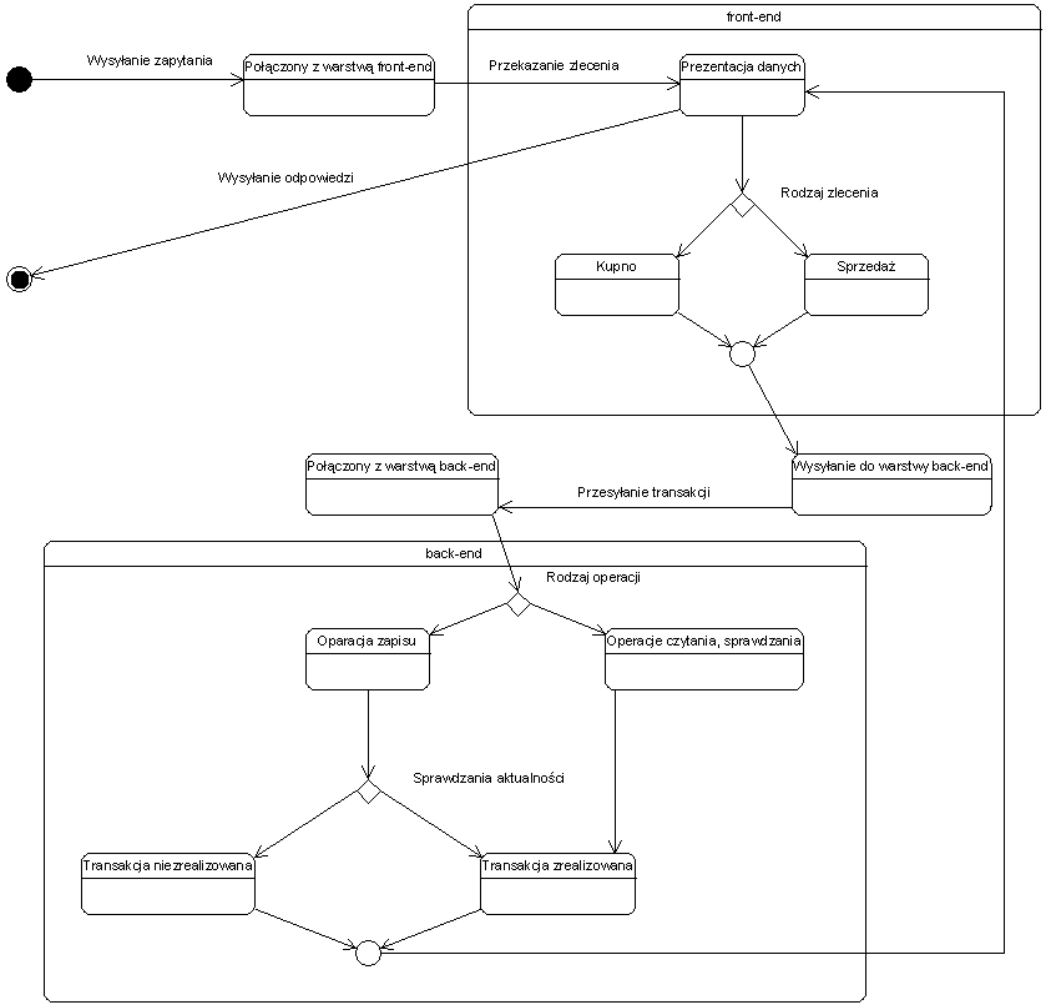
Poniżej zaprezentowano najważniejsze, z punktu widzenia rozpatrywanego problemu, diagramy UML dla uproszczonej giełdy internetowej. Centralna klasa to *Giełda*. Związana jest z nią instancja klasy *Sesja*. Konstruktor klasy *Sesja* tworzy trzy nowe instancje klas dziedziczących z *System notowań*, w którego skład wchodzi *System notowań ciągłych*. System działając w trybie ciągłym, co pewien interwał czasu przeprowadza aktualizację swojego stanu, której zadaniem jest modyfikacja oferty systemu. O ile obsługa i aktualizacja oferty giełdy odbywa się co pewien okres czasu, o tyle interakcja klienta (*Inwestor - Inwestor indywidualny*) z giełdą zachodzi już w czasie rzeczywistym. Do zadań klienta należy głównie przeglądanie akcji i składanie zapytań (*Zlecenie*). Graficzną interpretację w postaci modelu klas przedstawiono na rysunku 3.2:

- *Giełda* - główna klasa systemu zawierająca nazwę identyfikującą giełdę. Prowadzi notowania oraz powiązana jest z dostępnymi akcjami.
- *Sesja* - zawiera notowania ciągłe.
- *System notowań* - zawiera datę notowania i jego aktualny stan sesji.
- *Zlecenie* - zawiera się w *Koszyku zleceń* i posiada określony typ kupno lub sprzedaż. Pozostałe pola opisują parametry zlecenia. Operuje na pewnej grupie/pakiecie akcji i jest wysyłane na giełdę.
- *Akcja* - reprezentuje oferowane akcje spółek. Opisane są one przez nazwę spółki, aktualną cenę wraz z datą jej osiągnięcia oraz cenę otwarcia i zamknięcia w danym dniu oraz pulę dostępnych akcji.
- *Portfel* - zawiera akcje posiadane przez klienta w wyniku realizacji zlecenia.
- *Inwestor* - reprezentuje klienta giełdy. Każdy klient identyfikowany jest przez nazwę, posiada *Konto inwestycyjne*, do którego *Portfel* dostarcza określone akcje.




Rys. 3.2 Diagram klas uproszczonej giełdy internetowej

Spośród diagramów stanu wybrano diagram aktualności zapytania obejmujący obsługę zlecenia (Rys. 3.3). Prezentuje on etapy w jakich znajdują się zlecenia klienta w czasie przebywania w ISIROSO. Zaobserwować można sprawdzanie aktualności składanego zlecenia.



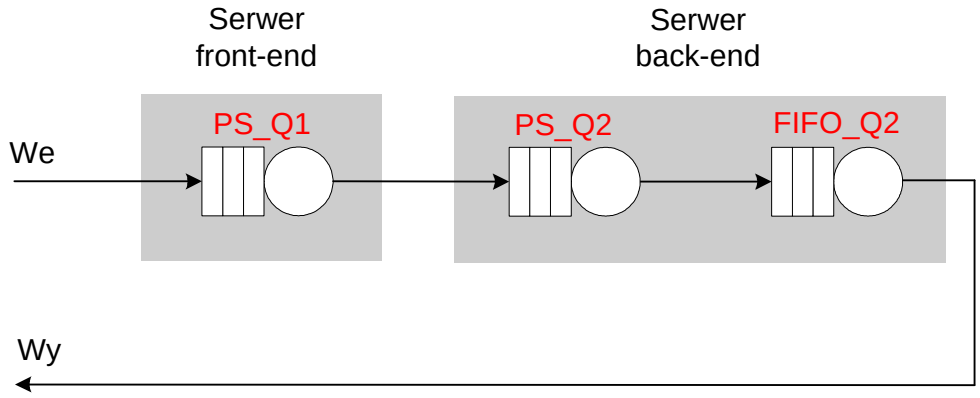

Rys. 3.3 Diagram stanu aktualności zlecenia uproszczonej giełdy internetowej

Po zaprezentowaniu diagramów UML, obrazujących budowę i zachowania internetowego systemu giełdowego, można jednoznacznie powiedzieć, że w systemie tym zapytania ulegają dezaktualizacji. Widać to dokładnie na przedstawionym diagramu stanu (Rys. 3.3). Szybkość realizacji zapytań klienta nie zależy już tylko i wyłącznie od możliwości jakie oferuje system. Konieczność przetworzenia zapytań, które straciły aktualność poprzez zmianę oferty systemu, wydłuża czasz realizacji pozostałych zapytań. Wybrane diagramy systemu uproszczonej giełdy internetowej można znaleźć w dodatku (por. dodatek).

### 3.2.3 PODSTAWOWY MODEL KOLEJKOWY

Podstawowy model warstwowy (model I) ISIROSO w postaci kolejkowej [87], składający się z dwóch warstw - *front-end* i *back-end*, przedstawiony został na rysunku 3.4. Do zamodelowania elementów warstw systemu użyto otwartej sieci kolejkowej, składającej się z systemów kolejkowych:

- PS - modeluje jednostkę przetwarzającą węzła. Zapisać go można jako M/M/1/PS/$\infty$ (rozkład wykładniczy strumienia zgłoszeń/rozkład wykładniczy procesu obsługi/jedno stanowisko obsługi zapytań/regulamin kolejki PS/nieskończona liczba zapytań w kolejce).
- FIFO - modeluje podsystem dyskowy serwera. Zapisać go można jako M/M/1/FIFO/$\infty$ (rozkład wykładniczy strumienia zgłoszeń/rozkład wykładniczy procesu obsługi zapytań/jedno stanowisko obsługi/regulamin kolejki FIFO/nieskończona liczba zapytań w kolejce).




Rys. 3.4 Kolejkowy model podstawowy systemu internetowego - model I

Serwer warstwy *front-end* (Rys. 3.4) modelowany jest przy pomocy systemu kolejkowego PS_Q*1* - reprezentującego jednostkę przetwarzającą, a serwer *back-end* (Rys. 3.4) poprzez kolejkę PS_Q*2* - modelującą jednostkę przetwarzającą serwera, współpracującą z kolejką FIFO_Q*2* - modelującą dysk serwera [50]. Proces napływania zapytań do systemu modelowany jest przy pomocy rozkładu wykładniczego z parametrem $\lambda$, a proces obsługi zapytań według rozkładu wykładniczego z parametrem $\mu$. Zasady tworzenia modelu kolejkowego systemu internetowego zaczerpnięto z książki [48] i prac [18, 50, 51] oraz własnych doświadczeń [86, 88, 89].

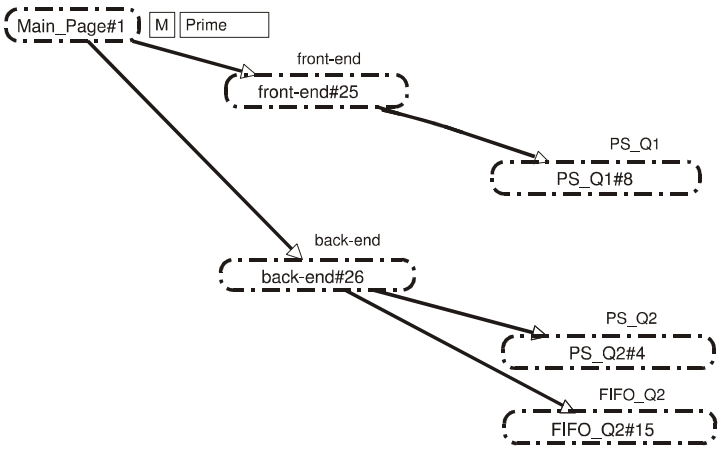
### 3.2.4 MODEL TCPN

Wielu analityków czuje potrzebę zastosowania symulacji modeli systemów rzeczywistych. Powodem jest potrzeba zrozumienia oraz przewidzenia ich zachowania.

Zaproponowane podejście można traktować jako rozwinięcie rozwiązań wprowadzonych w publikacji [51]. Ideę kolejkowych sieci Petriego przeniesiono na, szerzej rozpowszechniony formalizm czasowych kolorowanych sieci Petriego [35, 45, 66, 95, 106, 107]. Do utworzenia modeli systemów kolejkowych zastosowano wbudowany w narzędzie Design/CPN [35] język

funkcyjny CPN ML (ang. *CPN Modelling Language*) [133], umożliwiający naturalne definiowanie typowych struktur danych i operacji na nich [84]. Korzystając z cech sieci TCPN zaimplementowanych w pakiecie Design/CPN [35, 45, 66], opracowano zasady przepływu znaczników w sieci, zbliżone do zaproponowanych w uogólnionych kolorowanych stochastycznych sieciach Petriego [9]. Znaczniki reprezentują zapytania, które są przetwarzane w poszczególnych warstwach systemu. Uzyskano ostatecznie wersję narzędzia programowego, które w sposób symulacyjny może odwzorowywać czasowe zachowanie sieci kolejkowych. Dzięki zastosowaniu modułu analizy wydajnościowej sieci TCPN [66], możliwe jest również efektywne przechwytywanie i analiza danych dla utworzonych modeli.

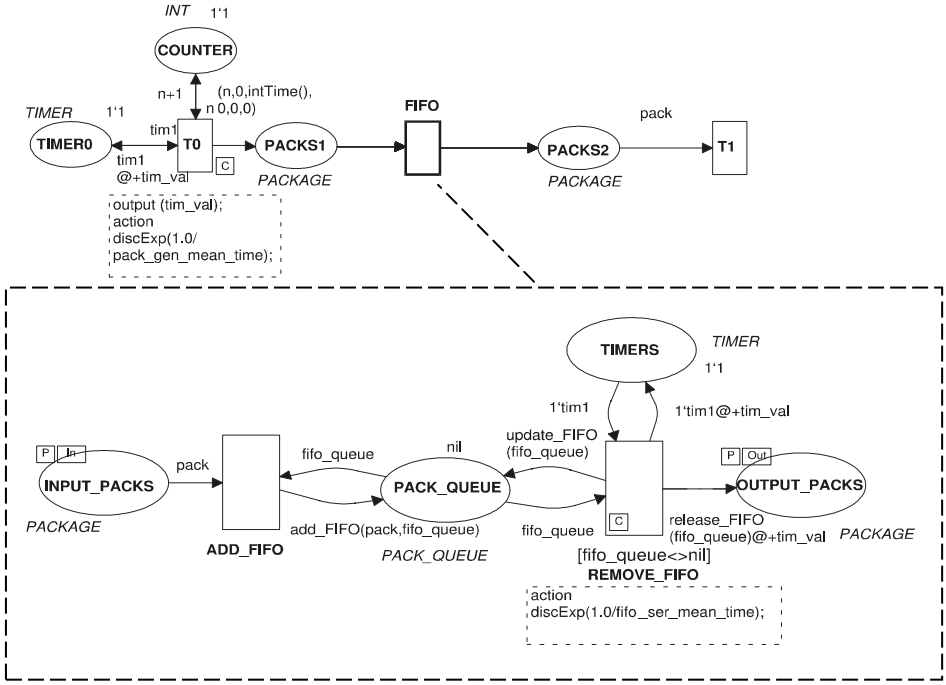
Opracowano wzorce projektowe kilku systemów kolejkowych najczęściej stosowanych do reprezentacji właściwości komponentów warstwowych systemów internetowych (FIFO i PS) [84]. Każdy wzorzec projektowy jest odpowiednią siecią TCPN („podstroną"), którą, stosując mechanizmy hierarchizacji sieci (Rys. 3.5), można włączyć do modelu systemu jako „podstawiane" przejście [45]. Odwzorowywanie własności sieci kolejkowej polega na konstrukcji pewnej sieci TCPN, w której część przejść definiowana jest jako systemy kolejkowe.



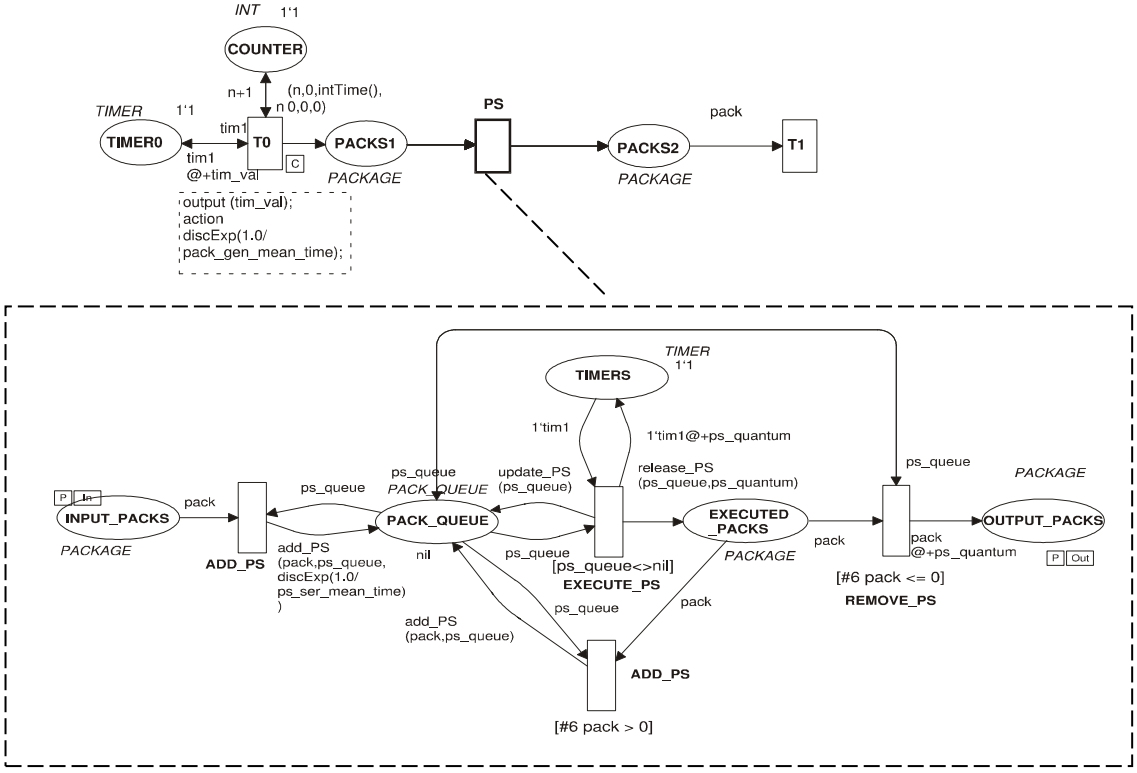

Rys. 3.5 Hierarchia „podstron" modelu TCPN systemu podstawowego - model I

Ostateczny model czasowej kolorowanej kolejkowej sieci Petriego uzyskuje się przez dołączenie do wymienionych przejść odpowiednich „podstron", zawierających implementacje określonego systemu kolejkowego. Pojedyncze systemy kolejkowe FIFO i PS zweryfikowano za zgodność z modelami analitycznymi przy wykorzystaniu równań Chapmana-Kołmogorowa [39] dla średnich długości kolejek i średniego czasu obsługi (Tab. 2.1). Dodatkowo ich poprawność potwierdzono [84] dzięki porównaniu z wynikami dla pojedynczych systemów kolejkowych przedstawionych również w [48].

Na rysunku 3.6 zobrazowano realizację systemu kolejkowego typu M/M/1/FIFO/$\infty$. Na rysunku 3.7 zaprezentowano z kolei realizację systemu kolejkowego typu M/M/1/PS/$\infty$. Pełny opis modelu wymaga wskazania definicji kolorów i funkcji w języku CPN ML powiązanych z elementami sieci (umieszczanych w węźle deklaracyjnym sieci): FIFO i PS (por. dodatek).




Rys. 3.6 Sieć TCPN odwzorowująca system kolejkowy M/M/1/FIFO/∞




Rys. 3.7 Sieć TCPN odwzorowująca system kolejkowy M/M/1/PS/∞

Prezentowany model umożliwia symulowanie zachowania systemu kolejkowego przez „wykonywanie” sieci. Dla FIFO na głównej „stronie” (górna część rysunku 3.6) zdefiniowano proces wejściowy systemu kolejkowego (wykonanie przejścia *T*0). Znaczniki generowane zgodnie z wykładniczym rozkładem częstotliwości napływania (traktowane jako indywidualne zapytania) gromadzone są w postaci czasowego wielozbioru w miejscu *PACKS*1, a następnie wprowadzane do systemu kolejkowego oznaczanego na „stronie” jako przejście „podstawiane” *FIFO*. Po opuszczeniu kolejki znaczniki są usuwane z sieci przez „wykonywanie” przejścia *T*1. W momencie generowania każdego znacznika, następuje między innymi wpisanie w jego strukturę czasu utworzenia, pozwalające na późniejszą analizę czasu odpowiedzi na zapytania.

W celu zapewnienia możliwości modelowania większych systemów w trakcie opracowania modelu sieciowego, uwzględniono możliwość budowania struktury hierarchicznej sieci. Warstwa nadrzędna wydziela warstwy systemu internetowego reprezentowane przez odpowiednie przejścia „podstawiane”. Każde z „podstawianych” przejść jest powiązane z odpowiednim modelem systemu kolejkowego. Rysunek 3.8 zawiera, główną oraz podrzędne, „strony” sieci TCPN modelujące omówioną sieć kolejkową (Rys. 3.4). Przejście *T*0 wraz z otaczającymi je miejscami stanowi model procesu napływania zapytań. Warstwę *front-end* oraz warstwę *back-end* przedstawiono na głównej „stronie” sieci TCPN jako przejścia „podstawiane”, odpowiednio: *front-end* i *back-end*. Górna i dolna część rysunku 3.8 to „podstrony” powiązane z przejściem „podstawianym” *front-end* i *back-end*. Przejścia na wymienionej podstronie reprezentują systemy kolejkowe odpowiadające składnikom modelu przedstawionym na rysunkach 3.6 i 3.7.

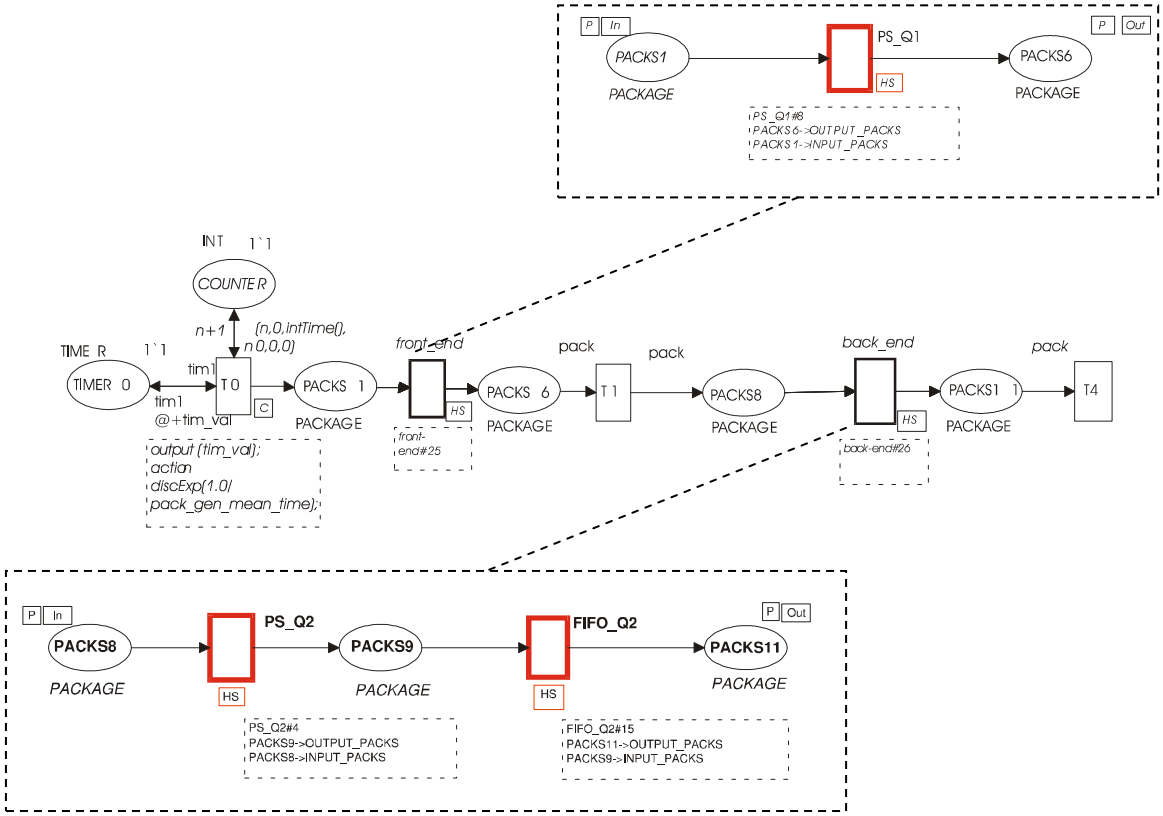
Elementy struktury sieci:

- miejsce oznaczające system wejściowy,
- miejsce oznaczające kolejkę (model w języku ML),
- miejsce definiujące stanowiska obsługi jako wielozbiór czasowych znaczników, każdy z symulacją jednego stanowiska obsługi,
- miejsce oznaczające wyjście z systemu kolejkowego,
- przejście dodające elementy do kolejki,
- przejście obsługujące wyjście z kolejki (pobiera elementy z kolejki i symuluje czas obliczeń poprzez nadawanie „stempli” czasowych znacznikom).

Parametry systemu kolejkowego:

- średnia wartość intensywności napływania zapytań,
- średni okres czasu obsługi dla kolejki o regulaminie FIFO,
- wartość przedziału czasu (*ps_quantum*), po którym następuje przełączenie pomiędzy obsługiwanymi zapytaniami dla kolejek o regulaminie PS (Podrozdz. 2.3.2),
- liczba stanowisk obsługi.

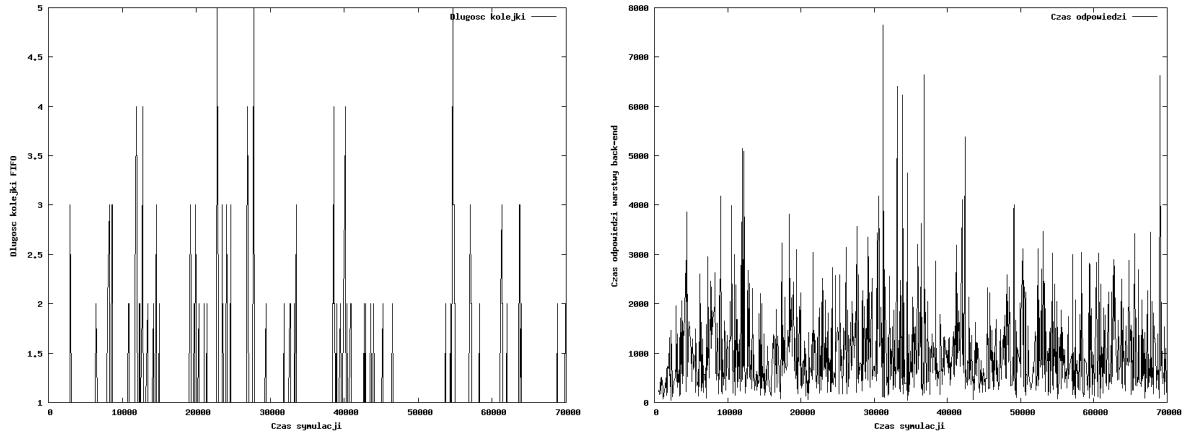
Jak wspomniano, podstawowy model I systemu internetowego klasy ISIROSO (Rys. 3.4) odwzorowano w postaci sieci TCPN (Rys. 3.8) skonstruowanej według uproszczonego modelu sieci kolejkowej dla warstwowego systemu internetowego. W rezultacie otrzymuje się „wykonywalny”, w sensie symulacji, model sieci kolejkowej. Generowane przez proces wejściowy znaczniki przenoszone są kolejno przez modele warstw.




Rys. 3.8 Model sieci TCPN - model I

Moduł wydajnościowy Design/CPN [66] umożliwia monitorowanie i w dowolnym miejscu modelu przechwytywanie wybranych elementów stanu sieci w chwili zaistnienia pewnego zdarzenia podczas symulacji i zapisywanie do plików informacji o np. aktualnej długości kolejek oraz czasie odpowiedzi. Wyniki po kolejnej obróbce można przedstawić na wykresach (por. dodatek), co pozwala wnioskować o zachowaniu systemu. Do prezentacji graficznej wyników użyto narzędzia *gnuplot*. Monitorowanie podanych parametrów umożliwia określenie, w jakim czasie system jest w stanie przetwarzać zapytania i czy jest zrównoważony.

Rysunek 3.9 zawiera przykładowe przebiegi uzyskane podczas symulacji omawianego modelu I z grupy ISIROSO (Rys. 3.4), dla parametrów: $\lambda = 5[1/s]$ i $\mu = 10[1/s]$ dla wszystkich stanowisk obsługi.

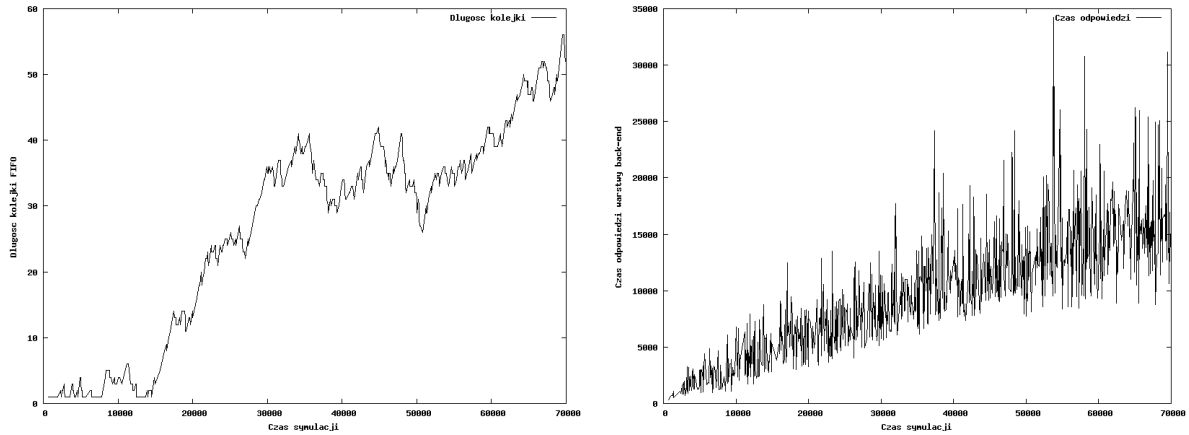



a) b)

Rys. 3.9 Symulacja TCPN modelu I: a) długość kolejki FIFO w drugiej warstwie, b) czas odpowiedzi warstwy back-end systemu - przypadek zrównoważony [84]

Eksperyment obejmował przedział czasu modelowego od 0 do 70000 jednostek. Kolejka FIFO należąca do drugiej warstwy (Rys. 3.9a) była zrównoważona. Jej maksymalna długość nie przekraczała wartości 5, a średnio wynosiła 1,3 zapytania. Średni czas odpowiedzi w warstwie *back-end* wynosi 1025 (Rys. 3.9b), przy czym maksymalna wartość czasu odpowiedzi wyniosła 7854 jednostek. Czasy nie wzrastają i oscylują wokół wartości średniej. W takim przypadku można mówić o zrównoważeniu systemu.

Na rysunku 3.10 przedstawiono przebiegi symulacji dla przypadku niezrównoważonego systemu, dla parametrów $\lambda = 70[1/s]$ i $\mu = 10[1/s]$ dla wszystkich stanowisk obsługi. Prezentowane wyniki dotyczą tej samej kolejki i czasu, co uprzednio i identycznego przedziału czasu symulacji. Długość monitorowanej kolejki (Rys. 3.10a) i średni czas odpowiedzi warstwy *back-end* (Rys. 3.10b) narastają w czasie. Na podstawie przedstawionych wykresów wnioskować można o konieczności wprowadzenia modyfikacji w strukturze systemu. Proponowane narzędzie programowe analizuje parametry w poszczególnych warstwach systemu internetowego co pozwala na wykazanie zrównoważenia systemu.




a) b)

Rys. 3.10 Symulacja TCPN modelu I: a) długość kolejki FIFO w drugiej warstwie, b) czas odpowiedzi warstwy back-end systemu - przypadek niezrównoważony [84]

Dysponując mechanizmami „przechwytywania czasowego stanu sieci" podczas symulacji, można dokonać doboru parametrów systemu w taki sposób, aby zachować zrównoważenie systemu.

### 3.2.5 MODEL SYMULACYJNY NA BAZIE PAKIETU CSIM

Pakiet CSIM jest zbiorem bibliotek funkcji napisanych w językach C oraz C++, które umożliwiają tworzenie zorientowanych na procesy modeli symulacyjnych wykorzystujących zdarzenia [113].

Modelowanie systemów internetowych w przypadku wykorzystania bibliotek CSIM nie wymaga tworzenia elementów składowych modelu symulacji. CSIM zawiera większość gotowych składników w postaci funkcji zarówno sieciowych, jak i elementów komputerowych. Budowa modelu symulatora, przy użyciu pakietu symulacyjnego CSIM, polega na zaimplementowaniu jego funkcji bazując na utworzonym wcześniej modelu kolejkowym. Ważnym elementem są parametry symulacji, które należy podać przed jej rozpoczęciem. Symulator zawiera biblioteki funkcji, które generują liczby losowe dla kilkudziesięciu rozkładów między innymi wykładniczego (Podrozdz. 2.3.2).

Kolejkowy model podstawowego systemu warstwowego (model I), (Rys. 3.4) zamodelowany przy użyciu bibliotek CSIM, został bardziej szczegółowo opisany w dalszej części przy okazji omawiania modelu CSIM dla struktur klastrowo-replikacyjnych. Wybrane szczegóły implementacji replikacji zaczerpnięto z prac M. Wiesmanna [119, 120, 121]. W dodatku zaprezentowano wybrane elementy aplikacji symulującej pakietu CSIM. Tu podano jedynie interpretację otrzymywanych przy jej pomocy wyników.

Po zakończeniu działania symulacji otrzymuje się pliki z raportami. Zawierają one tabele opisujące cały system, jak również jego poszczególne elementy np. (Lis. 3.1):

- długość kolejki,
- wykorzystanie zasobów,
- przepustowość,
- czas odpowiedzi,
- liczba wykonań.

Lis. 3.1 Raport zbiorczy po symulacji w CSIM (fragment) - dla parametrów: $\lambda = 100$ i $\mu = 100$ dla wszystkich stanowisk obsługi (całość por. dodatek)

```
              C++/CSIM Simulation Report (Version 19.0 for MS Visual C/C++)
                                  Queueing Network Model
                                Tue Oct 17 12:08:49 2006
                           Ending simulation time:    100298.99
                           Elapsed simulation time:   100298.99
                           CPU time used (seconds):     2870.97

     FACILITY SUMMARY
Facility   service       utilization  throughput   queue length   response time
Name       disc
PS_Q1      prc_shr             99.00        0.00        1960.89           10240
PS_Q2      prc_shr              0.01        0.09           0.00              16
FIFO_Q2    fifo                 0.01        0.09           1.00              18
```

```
      PROCESS CLASS SUMMARY
 id         name        number      lifetime  hold count   hold time   wait time
 0          prc_shr          10002     22.05         1.99       11.02       11.02
 1          prc_shr          30000      0.66         0.66        1.60        0.93
 2          fifo             20000      1.50         1.50        5.48        3.98
```

Prezentowany listing 3.1 przedstawia zbiorcze statystyki, parametrów wymienionych wcześniej, a w tym przykładzie są to: zasoby i procesy. System obciążony jest natężeniem zapytań wynoszącym 100[zapytań/s] ($\lambda = 100[1/s]$), a obsługa dla wszystkich kolejek jest taka sama i wynosi 100[zapytań/s] ($\mu = 100[1/s]$).

Po otrzymaniu wyników możliwe jest wyznaczenie charakterystyk wydajnościowych systemu lub zebranie wyników w tabelach, co znacznie ułatwia późniejszą analizę. W CSIM brak, podobnie jak CPN/Design, narzędzia do graficznej interpretacji wyników.

Modele CSIM wykorzystano do sprawdzenia wyników TCPN. Weryfikacja ta polegała na porównaniu wybranych parametrów systemów. Przykładowe wyniki długości kolejek dla modelu I (Rys. 3.4) przy różnych wartościach parametru $\lambda$ przedstawiono w tabeli 3.1. Zaobserwować można tu jedynie kilku procentowe odchylenia pojawiające się w wyniku drobnych różnic w samej implementacji modeli.

Tab. 3.1 Przykładowe wyniki symulacji dla TCPN i CSIM dla modelu podstawowego - średnie długości kolejek dla trzech różnych obciążeń

| $\lambda[1/s]$ | Rodzaj / Kolejka | Model TCPN | Model CSIM | Błąd [%] |
|---|---|---|---|---|
| 100 | PS1_1 | 2149,61 | 1960,89 | 8,8 |
| | PS2_1 | 0,00 | 0,00 | 0,0 |
| | FIFO2_1 | 1,00 | 1,00 | 0,0 |
| 300 | PS1_1 | 14950,90 | 16001,97 | -7,0 |
| | PS2_1 | 1,52 | 1,63 | -20,3 |
| | FIFO2_1 | 1,25 | 1,43 | -14,3 |
| 500 | PS1_1 | 45727,99 | 40034,34 | 12,4 |
| | PS2_1 | 6,85 | 8,90 | -29,9 |
| | FIFO2_1 | 5,96 | 7,00 | -17,3 |

Wyniki z tabeli 3.1 przedstawiają również orientacyjny błąd jaki jest pomiędzy zmierzonymi wartościami średnich długości kolejek dla modeli TCPN i CSIM. Przy jednakowym czasie symulacji. Znak określa jedynie charakter błędu a wartość liczona jest na podstawie prostej proporcji [48]:

$$BLAD[\%] = \frac{WARTOSC_1 - WARTOSC_2 * 100[\%]}{WARTOSC_1} \tag{3.9}$$

## 3.3 PODSUMOWANIE ROZDZIAŁU

Rozdział rozpoczyna się przedstawienie różnych form e-biznesu. Jedną z nich jest handel elektroniczny, do którego należą systemy internetowe. Spośród tych systemów wyłania się grupa, w której działaniu uwidacznia się dynamicznie zmieniająca się oferta systemu (ISIROSO). Przedstawiony podział daje obraz różnych rodzajów systemów internetowych, z których wybrano, do dalszej analizy, system giełdowy jako przedstawiciela ISIROSO. Na potwierdzenie dynamiki zmian oferty systemu zaprezentowano model formalny (Podrozdz. 3.2.1), w którym przedział czasu, po jakim pojawia się nowa oferta systemu, jest porównywalny lub krótszy od czasu, jaki klient potrzebuje, aby zareagować na zaistniałą sytuację i wysłać kolejne aktualne zapytanie. W takim przypadku może ujawnić się brak aktualności wysyłanego zlecenia, co ilustruje diagram stanu modelu UML (Podrozdz. 3.2.2).

Wykazano, że opisana w rozdziale uproszczona giełda internetowa z notowaniami ciągłymi jest reprezentantem klasy ISIROSO.

Wydajność systemów internetowych jest funkcją wielu zmiennych, których zachowanie jest trudne do przewidzenia. Nie stworzono jeszcze gotowych standardów i przepisów a istniejące rozwiązania dotyczą zazwyczaj ogólnych przypadków. Aby uzyskać wiarygodne rezultaty, należy wykorzystać różne techniki zarówno teoretyczne, jak i praktyczne. W rozdziale zaprezentowano i przeanalizowano możliwości modelowania wydajności systemów internetowych bazując na modelach czasowych kolorowanych sieci Petriego (Podrozdz. 3.2.4) i symulacji przy użyciu pakietu CSIM (Podrozdz. 3.2.5), wynikających bezpośrednio z modelu kolejkowego - węzły systemów jako kolejki zapytań. Zastosowane podejście polega na zaproponowaniu modelowania systemów kolejkowych w oparciu o formalizm sieci TCPN i opracowaniu narzędzia programowego „wykonywalnych" modeli sieci kolejkowych przy użyciu pakietu Design/CPN. Drugi nieformalny sposób (CSIM) polega na zamodelowaniu tej samej sieci kolejkowej przy wykorzystaniu gotowych funkcji do modelowania wydajności systemów. Przedstawiono zasady posługiwania się skonstruowanymi modelami do analizy wydajności ISIROSO. Dzięki temu, że do symulacji używa się aplikacji napisanych w języku np. C++, jest ona również zrozumiała dla szerszego kręgu odbiorców[4]. Wstępne wyniki [84] wskazują na poprawność budowanych modeli. Do wykazania poprawności utworzonych modeli należy wykorzystać wyniki badań eksperymentalnych przeprowadzonych na zbudowanym systemie.

Zaproponowany model formalny i prezentowany model w języku UML, bazujące na założeniach WGPW, zostały użyte do budowy uproszczonego referencyjnego systemu giełdowego (Rozdz. 6). Modyfikacje modelu podstawowego (model I) zaprezentowano w rozdziale 4 wraz z odpowiadającymi im modelami formalnymi i nieformalnymi. Analizę wyników modelowania potwierdzającą poprawność stosowanych metod zaprezentowano w kolejnych rozdziałach.

[4] Najnowsza (2006r.) wersja bibliotek dostępna jest dla środowiska Javy.

# 4 ZAGADNIENIE DOBORU ARCHITEKTUR DLA ISIROSO

W rozdziale uściślono pojęcia klastra i replikacji. Przedstawiono, w postaci kolejkowej, własne propozycje architektur ISIROSO z uwzględnieniem klastrowania węzłów w warstwach (model klastra warstwy *front-end* i model klastrów obu warstw) i replikacji bazy danych (model klastra z replikacją). Omówiono sposoby budowy, symulacji i interpretacji wyników symulacji modeli formalnych wykorzystując narzędzie Design/CPN i symulacyjnych przy użyciu CSIM.

Celem rozdziału jest przedstawienie efektywnych architektur, umożliwiających realizację ISIROSO z klastrami i replikacją bazy danych. Klastry zostały wybrane ze względu na możliwość rozdzielenia - zapytań przybywających do systemów - na większą liczbę węzłów przetwarzających, co umożliwi szybsze przetwarzanie w warstwach. Zastosowanie replikacji wynika z możliwości przyspieszenia obsługi poprzez zrównoleglenie przetwarzania w bazie danych warstwy *back-end*.

## 4.1 ARCHITEKTURY SYSTEMÓW INTERNETOWYCH

Prace nad rozwojem architektur systemów internetowych trwają nieustannie. Wydzielenie warstw jest dość oczywiste, jednak przetwarzanie klastrowo-równoległe czy replikacja danych, to metody stosowane i rozwijane od niedawna w odniesieniu do systemów internetowych. Zastosowanie klastrów i replikacji w warstwach systemu giełdowego pociąga za sobą dodatkową złożoność. Jednak dzięki zastosowaniu odpowiedniej liczby i połączeń węzłów możliwe jest poprawienie czasu odpowiedzi systemów internetowych [48].

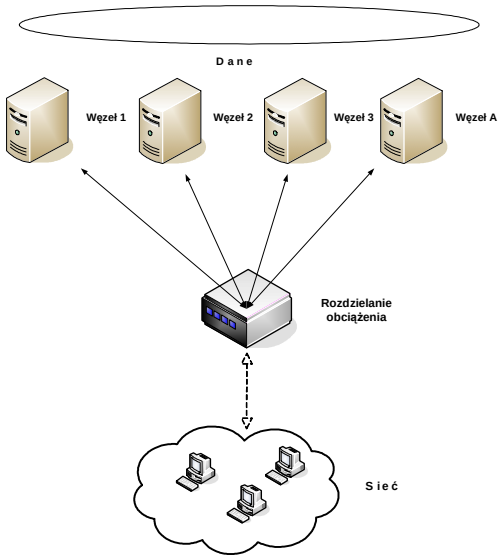
Podsumowując, klaster i replikacja pozwalają na rozdzielenie obliczeń i wprowadzają równoległe przetwarzanie danych. Stosowanie innych rozwiązań nie jest tak efektywne, jak stosowanie i łączenie tych dwóch metod [86].

### 4.1.1 KLASTRY

Dynamiczny rozwój architektur procesorów i towarzyszący temu zjawisku spadek cen umożliwiły tworzenie wydajnych systemów równoległych, stanowiących dobrą alternatywę dla drogich superkomputerów. Klastry znalazły zastosowanie w średnich i dużych przedsiębiorstwach oraz instytucjach naukowych. Dzięki swoim właściwościom są używane w złożonych systemach internetowych. Stosuje się je do zapewnienia dostępności, skalowalności i wydajności w warstwie prezentacyjno-aplikacyjnej [117] oraz wydajności i niezawodności w warstwie bazy danych [60, 61].

Celem zastosowania technologii klastrowej jest budowa skalowalnych i niezawodnych systemów wysokiej dostępności i wydajności [21]. Spośród wielu przykładów klastrów systemowych na szczególne zainteresowanie zasługują klastry linux'owe, a wśród nich: wydajnościowe (np. Beowulf), wyrównujące obciążenie (np. Mosix), internetowe (np. LVS, Piranha), magazynująco-pamięciowe (np. Sistina GFS (ang. *Global File System*)), bazodanowe (np. Oracle z RAC (ang. *Real Application Cluster*), MySQL Cluster) oraz wysokiej dostępności (np. Heartbeat).

Na uwagę, w tym przypadku, zasługują klastry internetowe, a szczególnie LVS [143]. Projekt ten łączy w sobie wysoką dostępność - HA (ang. *High Availability*) jak i wysoką wydajność - HP (ang. *High Performance*) (Rys. 4.1).




Rys. 4.1 Poglądowa struktura klastra internetowego - LVS

Podstawową cechą klastrów LVS jest „sprawiedliwość" równoważenia obciążenia na wiele serwerów. Nie tylko obniża to koszty utrzymania łączy, ale również odwiedzający serwis WWW są obsługiwani znacznie szybciej. Naturalna dla klastrów nadmiarowość w przypadku awarii pojedynczych serwerów zapewnia nieprzerwaną pracę. Stosowane wysoko wydajne maszyny, często mają niewspółmiernie wysoką cenę w stosunku do oferowanych usług. Stosowane w LVS komputery klasy PC, zapewniają możliwość korzystania z architektury HA i HP w środowisku laboratoryjnym, bez konieczności ponoszenia wygórowanych kosztów.

W poszczególnych warstwach mogą być uruchamiane różne typy klastrów, samodzielnie lub w połączeniu z innymi mechanizmami poprawiającymi parametry czasowe systemów. Ich sposób dzielenia obciążenia pozwala na „równomierną" dystrybucję zapytań. Mechanizm ten może obsługiwać praktycznie każdy rodzaj usług. Przykładowo w przypadku HTTP (ang. *Hypertext Transfer Protocol*) pociąga to za sobą jedynie konieczność dostępu do lustrzanego systemu plików, który zawiera identyczne dane.

### 4.1.2 REPLIKACJA BAZ DANYCH

W rozproszonych systemach baz danych, dane przechowywane są w skończonej liczbie lokalizacji/replik (ang. *site*), które często znajdują się w geograficznie rozproszonych miejscach. Dla wielu systemów, np. z zakresu bankowości czy telekomunikacji, rozproszone bazy danych stanowią bardziej naturalny model niż monolityczne systemy zcentralizowane. Niektóre komercyjne bazy danych (np. [156]) zapewniają obsługę rozdzielania danych na poziomie tzw. „silnika" bazy (ang. *database engine*), a nowe technologie komunikacyjne pozwalają na coraz większy stopień rozproszenia danych [15, 76].

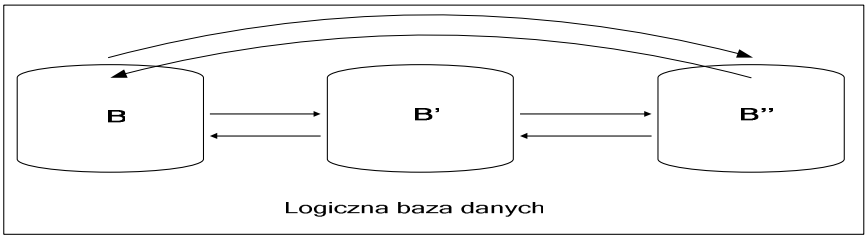
Wraz z rozwojem systemów baz danych pojawiło się bardziej zaawansowane rozwiązanie tj. replikacja. Replikacja oznacza powielanie, a więc tworzenie kopii (replik) oraz utrzymywanie

jednakowych danych w środowisku wielu baz rozmieszczonych w różnych węzłach (Rys. 4.2). Zmiany w jednej bazie są dokonywane lokalnie, a następnie przekazywane do każdej z odległych lokalizacji. Replikacja jest zatem mechanizmem, który tworzy i zarządza zespołem wielu kopii danych, na których pracują użytkownicy. Pojęcie replikacji powiązane jest z rozpraszaniem danych, jednak w przypadku:

- bazy rozproszonej, pojedyncza tabela znajduje się tylko w jednej lokalizacji,
- replikacji, kopie pojedynczej tabeli znajdują się w każdej lokalizacji.

Utrzymanie replikowanych danych jest więc ściśle powiązane z komunikacją pomiędzy lokalizacjami, a zarządzanie procesem replikowania ma znaczący wpływ na wydajność całych systemów. Zarządzanie replikowaniem polega na decydowaniu, kiedy i gdzie należy umieścić kopie fizyczne odpowiadające logicznym fragmentom danych oraz kiedy i jak je uaktualniać celem uzyskania akceptowalnej jednorodności i aktualności danych [31, 42]. Istnieją dwa typy replikacji:

- replika „kopii głównej" (ang. *master-slave*) zawiera pełną lub częściową kopię obiektów węzła docelowego (ang. *target master*) w danym momencie czasu. Pozostałe węzły są podrzędne w stosunku do węzła głównego.
- replikacja lokalizacji „nadrzędnych" (ang. *multi master*) pozwala na równe uczestnictwo wszystkich lokalizacji w zarządzaniu. Każda lokalizacja jest lokalizacją „nadrzędną" i każda komunikuje się z pozostałymi lokalizacjami „nadrzędnymi". Aplikacje mogą dokonywać uaktualniania danych w dowolnej lokalizacji.




Rys. 4.2 Baza danych z replikacją

Dostęp do danych bazy ( $B$ ) odbywa się przez transakcje ( $Tr_i$ ). Transakcja reprezentuje logiczną całość operacji $ZOp_{Tr_i}$ (Wzór 3.5): czytania, zapisywania, sprawdzania, przerywania lub zakończenia, która powinna spełniać kryteria ACID (ang. *Atomicity Consistency Isolation Durability*) [120]:

- niepodzielność (ang. *atomicity*) oznacza, że transakcja zostanie doprowadzona do końca lub zostanie wycofana,
- spójność (ang. *consistency*) zapewnia, że skutkiem transakcji jest poprawny stan bazy danych,
- wyłączność (ang. *isolation*) oznacza, że wykonywanie dwóch równoległych transakcji jest zupełnie odseparowane od siebie,
- trwałość (ang. *durability*) oznacza nieodwracalność transakcji.

Zagadnienie replikacji nie jest łatwe, gdyż musi być zagwarantowany dostęp do danych spełniający warunki ACID. Replikacja dzięki swoim właściwościom ma zastosowanie w wielu dziedzinach, a szczególnie w systemach rozproszonych i w bazach danych. Stosowane techniki i mechanizmy są identyczne pod względem koncepcji, ale różnią się w sferze realizacji i działania.

W przypadku, gdy wszystkie dane są kopiowane do wszystkich lokalizacji (replikacja lokalizacji „nadrzędnych"), przeciwdziała to występującym błędom i brakom spójności danych a uaktualnienie dokonywane jest w zakresie transakcji (ang. *transaction boundaries*)). Dotyczy to replikacji „gorliwej" (ang. *eager replication*). Istnieją również metody replikacji, dla których spójność danych nie jest czynnikiem najistotniejszym a uaktualnienie dokonywane jest po zatwierdzeniu transakcji (ang. *transaction commit*)). Mówi się wtedy o replikacji „leniwej" (ang. *lazy replication*). W połączeniu z klastrami, z powodzeniem stosowane są również replikacje „kopii głównej". Jeżeli chodzi o funkcjonalność, to „gorliwa" replikacja lokalizacji „nadrzędnych" jest pożądaną formą replikacji dla klastrów. Zapewnia ona optymalny rozkład obciążenia, dostęp do danych oraz gwarantuje spójność i tolerowanie błędów [83].

**Definicja 4.1.**

Replikowana baza danych jest zbiorem baz danych [117], przechowujących kopie tych samych elementów danych. Jeżeli zatem $B'=\{b_1',...,b_N'\}$ będzie kopią bazy danych $B$ (Def. 3.5), gdzie $B'=B$ i $b_k'$ jest $k$-tym elementem bazy danych $B'$, to operacje ze zbioru $ZOp_{Tr_i}$ (Wzór 3.5) w replikowanej bazie danych, dokonywane są na elementach $b_k$ lub $b_k'$.

Wyróżnia się element danych $b_k$ i jego fizyczną kopię $b_k'$, znajdującą się w różnych lokalizacjach. Mogą być realizowane operacje transakcji $Tr_i$ w dwóch szczególnych przypadkach dla:

- replikacji „kopii głównej":

$$Op_i^j \in \{czytaj(b_k)_i, sprawdz(b_k)_i, czytaj(b_k')_i, \\ sprawdz(b_k')_i, zapisz(b_k)_i, przerwij\_Tr_i, zakoncz\_Tr_i\} \tag{4.1}$$

- replikacji lokalizacji „nadrzędnych":

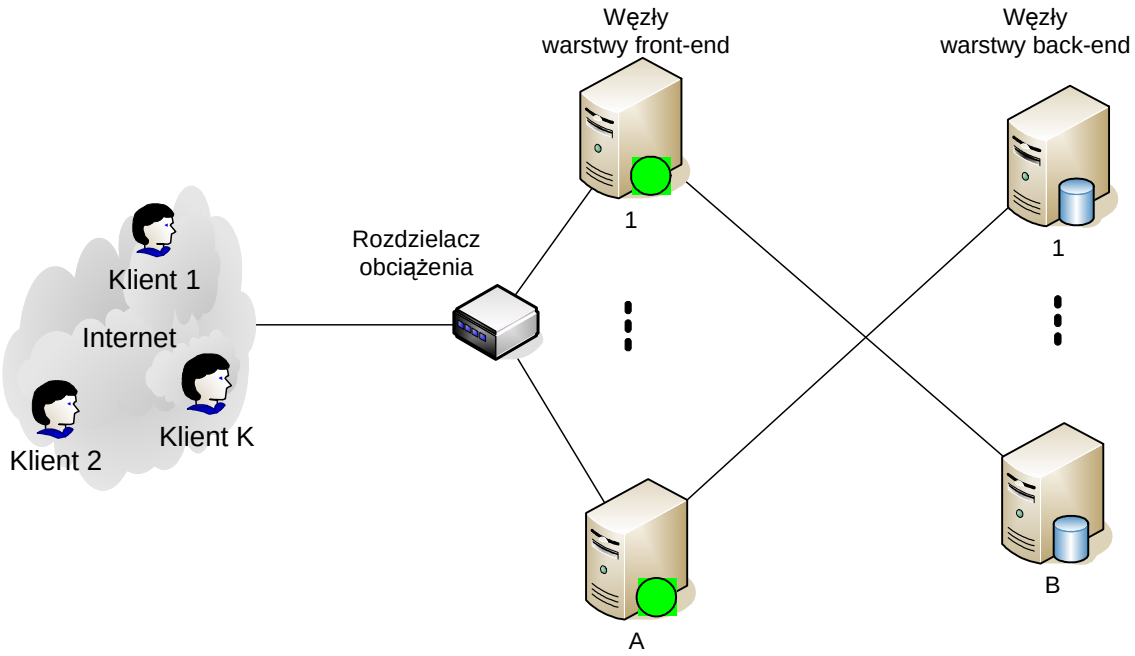
$$Op_i^j \in \{czytaj(b_k)_i, sprawdz(b_k)_i, czytaj(b_k')_i, \\ sprawdz(b_k')i_i, zapisz(b_k)_i, zapisz(b_k')_i, przerwij\_Tr_i, zakoncz\_Tr_i\} \tag{4.2}$$

Podsumowując, systemy z replikacją to: jedna logiczna baza danych, wiele fizycznych kopii bazy danych, pełna synchronizacja wszystkich kopii i zachowanie własności ACID [83]. Tak działająca baza danych może być elementem warstwy *back-end* systemów internetowych. W przypadku grupy ISIROSO replikacja pozwala na zrównoleglenie transakcji operujących na różnych danych oraz nie dopuszcza do realizacji transakcji, które modyfikują identyczne dane w bazie powodując ich wycofanie.

## 4.2 MODELE ISIROSO Z KLASTROWANIEM I REPLIKACJĄ BAZY DANYCH

W podrozdziale zaprezentowano utworzone modele kolejkowe dla ISIROSO wykorzystujące klastry i replikację bazy danych. W klastrach obciążenie rozkłada się na węzły, przetwarzające połączone szybką siecią [59, 82, 103]. Klastry zapewniają rozdzielanie ruchu, skalowalność i tolerowanie błędów. Replikacja wprowadza możliwość korzystania z większej liczby lokalizacji w czasie dokonywania transakcji.

Rysunek 4.3 przedstawia ogólny model systemów internetowych, składający się z dwóch zasadniczych warstw: *front-end (1, 2…A)*, realizującej funkcje prezentacji danych w środowisku klastrowym i *back-end (1, 2…B)*, realizującej funkcje bazy danych w środowisku replikacji bazy danych. Dyspozytor, równoważący obciążenie to dedykowane urządzenie lub oprogramowanie. Jednym z celów, który może realizować jest przekazywanie zapytań klientów zachowując wysoką jakość usług świadczonych przez systemy, określaną na podobieństwo terminu QoS, jako QoWS (ang. *Quality of Web Services*).




Rys. 4.3 Ogólny schemat systemów internetowych z klasterowaniem i replikacją

### 4.2.1 POGLĄDOWE MODELE KOLEJKOWE

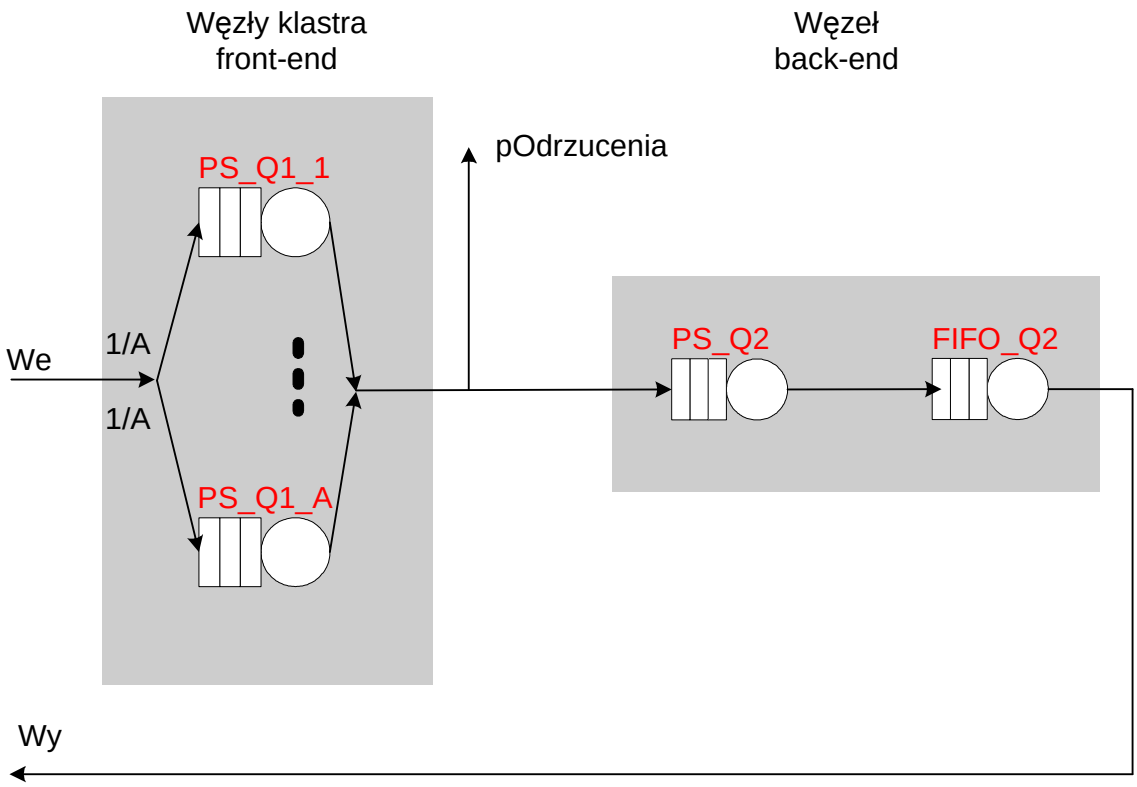
Do zaprezentowania modeli ISIROSO użyto, podobnie jak w rozdziale 3, otwartej sieci kolejkowej składającej się z systemów kolejkowych. Na podstawie zaprezentowanych poglądowych modeli omówiono budowy różnych struktur systemów internetowych. Przemieszczanie się zapytań w systemach definiowane jest na podstawie tras prawdopodobieństw.

Oprócz podstawowego modelu systemu giełdowego (model I) (Rys. 3.4), zaprezentowano w tym rozdziale trzy rozbudowane, również warstwowe, modele ISIROSO:

II. klaster w warstwie *front-end* (Rys. 4.4),
III. klaster w warstwie *front-end* i klaster w warstwie *back-end* (Rys. 4.5),
IV. klaster w warstwie *front-end* i klaster z replikacją w warstwie *back-end* (Rys. 4.6).

Celem tych prezentacji jest pokazanie stopnia złożoności architektury w kolejnych coraz bardziej rozbudowanych i szczegółowych modelach ISIROSO.

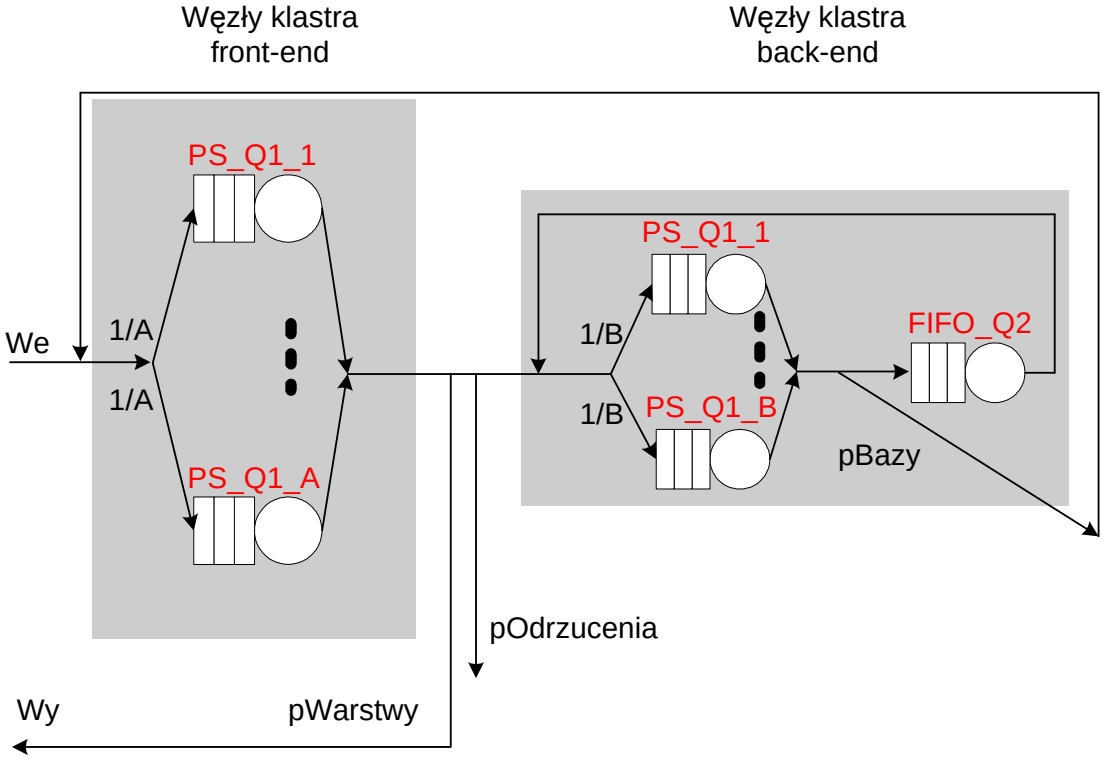
Model II przedstawiono na rysunku 4.4. Składa się on z dwóch warstw, gdzie w pierwszej z nich znajduje się klaster, modelowany jako połączone równolegle systemy kolejkowe (M/M/1/PS/∞). Zapytania trafiają do warstwy *front-end*, składającej się z klastra *A*-elementowego, kierowane są z prawdopodobieństwem *1/A* do jego węzłów. *1/A* modeluje mechanizm RR stosowany w procesie rozdzielania obciążenia (LB). *pOdrzucania* oznacza prawdopodobieństwo odrzucania zapytań po warstwie *front-end*. W badaniach przyjęto, podobnie jak w [125], że dyspozytor oraz lokalna sieć komputerowa wnoszą pomijalne opóźnienie w działanie całego systemu. Główne opóźnienia wnoszą serwery poszczególnych warstw.




Rys. 4.4 Kolejkowy model klastra warstwy front-end - model II

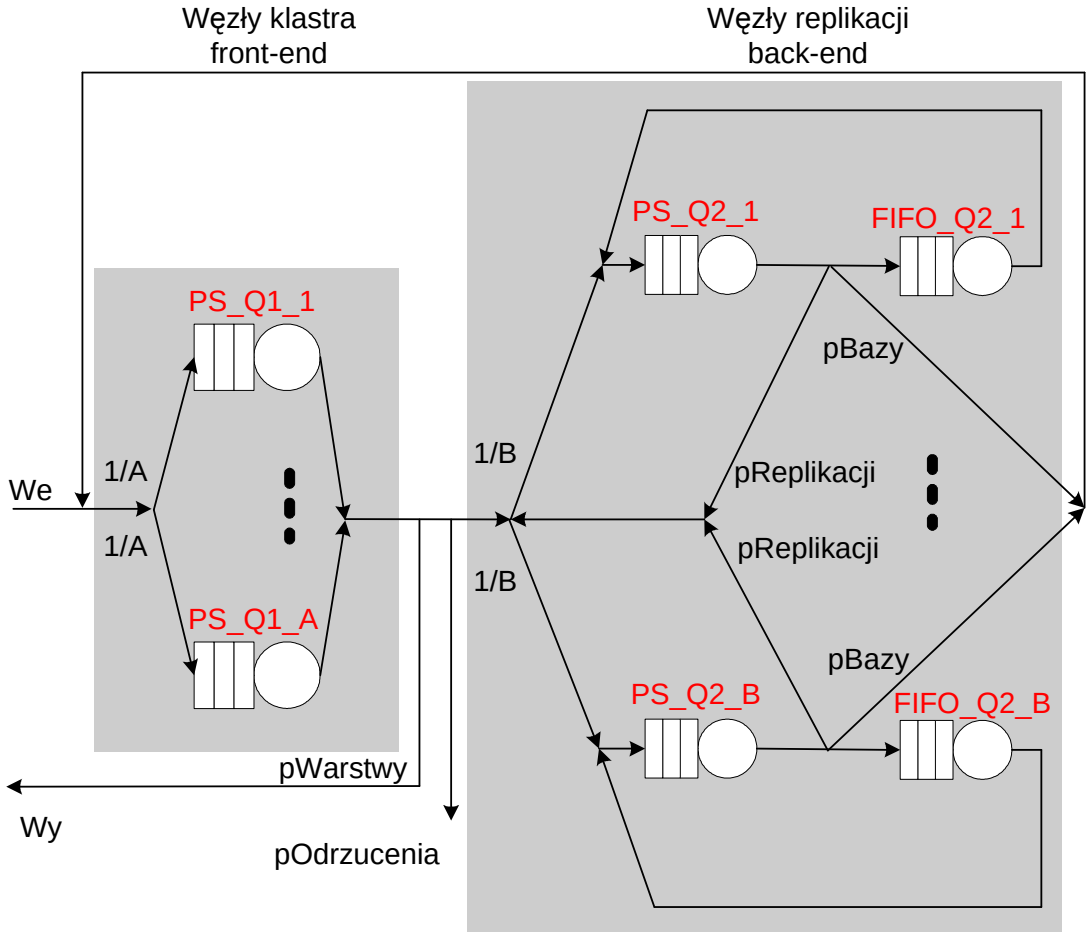
Model III składa się z dwóch warstw klastrów serwerów [87]. Większość opracowań dotyczących modelownia systemów internetowych upraszcza ten model do postaci pomijającej spotykane zjawisko wielokrotnego przekazywania zapytań pomiędzy warstwami [50], zastępując je jednokierunkowym strumieniem przekazywania zapytań [49, 88, 111]. Model III systemu w postaci kolejkowej, opracowany na podstawie ogólnego modelu (Rys. 4.3),

przedstawiono na rysunku 4.5. Zapytania klienta przesyłane są do wybranego węzła klastra *front-end* z prawdopodobieństwem *1/A*, gdzie ustawiają się w kolejce, aby otrzymać obsługę. Obsługa w jednostce przetwarzającej może zostać przerwana wielokrotnie, jeśli zapytania wymagają np. dostępu bazy danych. Za każdym razem, gdy to się zdarza, zapytania przesyłane są do warstwy *back-end*. W przypadku przesłania do bazy danych ustawia się do obsługi w jednym z węzłów *back-end* z prawdopodobieństwem *1/B*. Obsługa w jednostce przetwarzającej bazy danych może zostać przerwana w przypadku potrzeby dostępu podsystemu we/wy dysku bazy danych i po otrzymaniu obsługi, zapytania przesyłane są z powrotem do jednostki przetwarzającej bazy danych. Operacja ta może zostać powtórzona wielokrotnie. Po zakończeniu obsługi w warstwie *back-end*, zapytania odsyłane są do serwera *front-end* (*pBazy*). Zapytania mogą odwiedzać warstwę *back-end* wielokrotne podczas przetwarzania. Po zakończeniu obsługi w serwerze *front-end* zapytania odsyłane są do klienta (*pWarstwy*).




Rys. 4.5 Kolejkowy model klastrów obu warstw - model III

Model IV przedstawiono na rysunku 4.6. Przetwarzanie w tym modelu jest identyczne jak w modelu III klastrów obu warstw z tą jednak różnicą, że zapytania po otrzymaniu obsługi w jednostce przetwarzającej korzystają z zasobów dyskowych bazy danych a w przypadku zaistnienia konieczności replikacji przesyłane są do kolejnego węzła bazy (*pReplikacji*). Jeżeli nie zaistnieje żadna z tych dwóch sytuacji zapytania przesyłane są do warstwy *front-end* (*pBazy*). Tu podobnie jak w poprzednim przypadku po zakończeniu obsługi w serwerze *front-end* zapytania odsyłane są do klienta (*pWarstwy*). Należy wspomnieć, że modelowana replikacja może powodować również wycofanie transakcji. Zarówno dokonanie replikacji jak i odrzucenie realizacji transakcji modelowane jest w sposób uproszczony jako przekazanie zadania do kolejnej lokalizacji replikacji.




Rys. 4.6 Kolejkowy model klastra front-end i klastra z replikacją back-end - model IV

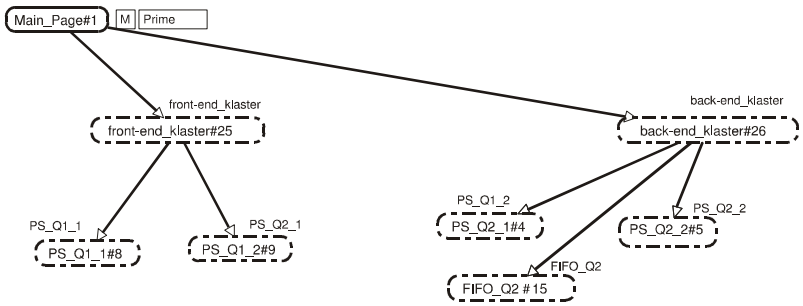
Zaprezentowane modele kolejkowe (Rys. 3.4, 4.4, 4.5 i 4.6) stanowią podstawę do dalszego modelowania przy użyciu Design/CPN i CSIM. Tworzone w dalszej części rozdziału modele wydajnościowe bazują więc na zaproponowanych modelach kolejkowych. Same modele kolejkowe nie będą analizowane a podane zostały w celu kompletnej prezentacji tworzonych modeli ISIROSO.

### 4.2.2 MODELE TCPN

Jak już wspomniano, sieci Petriego wysokiego poziomu cechują się wystarczającą „siłą ekspresji” dla modelowania i analizy złożonych systemów internetowych [48]. Na uwagę zasługuje duża „siła wyrazu” wynikająca z możliwości [95]:

- rozróżniania znaczników/zapytań i powiązania z komponentami sieci wyrażeń zapisanych w języku CPN ML [57],
- tworzenia sieci hierarchicznych - zwiększającej czytelność modelu dzięki technice budowania złożonych systemów z mniejszych komponentów [45],
- wyrażania wymagań wydajnościowych przez czasowe rozszerzenie sieci,
- konstruowania, wykonywania i opracowania formalnej analizy sieci poprzez dostępność narzędzi programowych [45, 134, 136].

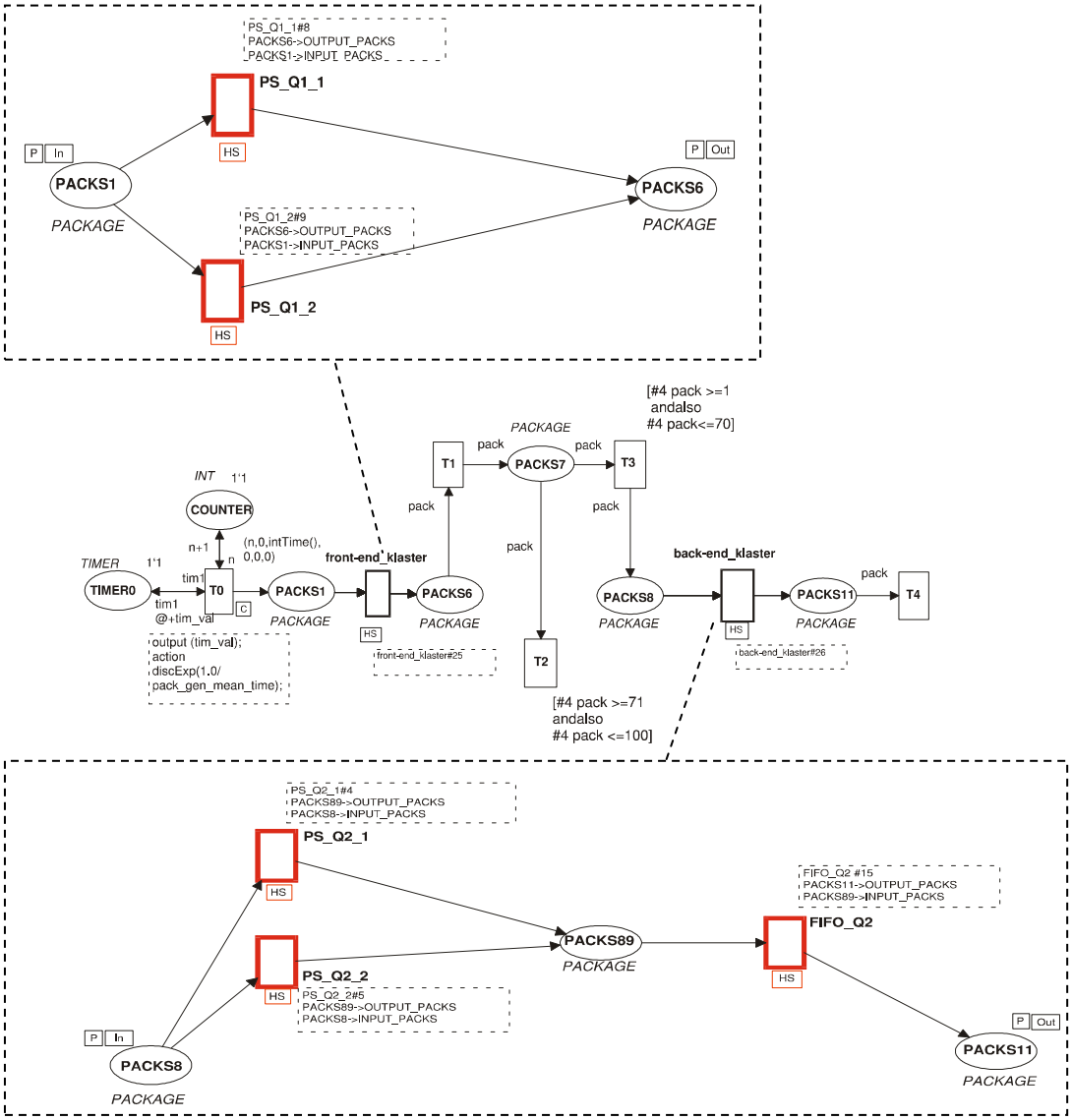
W podrozdziale przedstawione zostały modele ISIROSO w postaci czasowych kolorowanych sieci Petriego, na różnych poziomach złożoności. Pominięto omawianie najprostszych modeli - model I (Rys. 3.4) i model II - (Rys. 4.4), ponieważ wszystkie ich elementy występują w kolejnych bardziej złożonych modelach. Sieć kolejkową - model III - przedstawioną na rysunku 4.5 odwzorowano w postaci modelu czasowej kolorowanej sieci Petriego (Rys. 4.7). Przedstawiony na rysunku 4.8 model dotyczy jednokierunkowego przesyłania zapytań pomijając ich powrót do warstwy pierwszej oraz możliwość wielokrotnego odwoływania się do bazy danych.




Rys. 4.7 Hierarchia „podstron" modelu TCPN klastrów obu warstw - model III (przykład)

Rysunek 4.7 zawiera główną oraz podrzędne „podstrony" sieci TCPN, modelujące omówioną powyżej sieć kolejkową (Rys. 4.5) dla dwóch węzłów (PS_Q*1_1* i PS_Q*1_2*) klastra *front-end* (*A=2*), dwóch węzłów (PS_Q*2_1* i PS_Q*2_2*) i węzła FIFO_Q*2* klastra *back-end* (*B=2*).

Rozwinięciem rysunku 4.7 jest rysunek 4.8 na którym szczegółowo przedstawiono „podstrony" modelu TCPN. Przejście *T*0 (Rys. 4.8), będące generatorem zapytań, stanowi wraz z otaczającymi je miejscami model procesu wejściowego. Klaster serwerów warstwy *front-end* oraz klaster warstwy *back-end* przedstawiono na głównej „stronie" sieci TCPN jako przejścia „podstawiane", odpowiednio: *front-end_klaser* i *back-end_klaster*. Górna i dolna część rysunku 4.8 zawierają „podstrony" powiązane z przejściami „podstawianymi": *front-end_klaser* (przykładowo modelującą dwa węzły klastra prezentacyjno-aplikacyjnego) i *back-end_klaster* (przykładowo modelującą dwa węzły klastra dla bazy danych). Przejścia na wymienionych „podstronach" reprezentują systemy kolejkowe PS i FIFO opisane szczegółowo w podrozdziale 3.2.4. Znaczniki znajdujące się w miejscu *PACKS7* mogą służyć do „wykonywania" przejścia *T*2 lub pozostającego z nim w konflikcie przejścia *T*3. Wykonanie przejścia *T*2 usuwa dany znacznik z sieci (modelując utratę pakietu danych (*pOdrzucenia*)). Natomiast, jeśli dany znacznik pobrany zostanie z miejsca *PACKS7* na skutek wykonania przejścia *T*3, przekazywany jest do przetwarzania w drugiej warstwie systemu. Funkcje dozorów powiązane z wymienionymi przejściami określają proporcje pomiędzy zapytaniami odrzucanymi a pozostającymi w sieci.

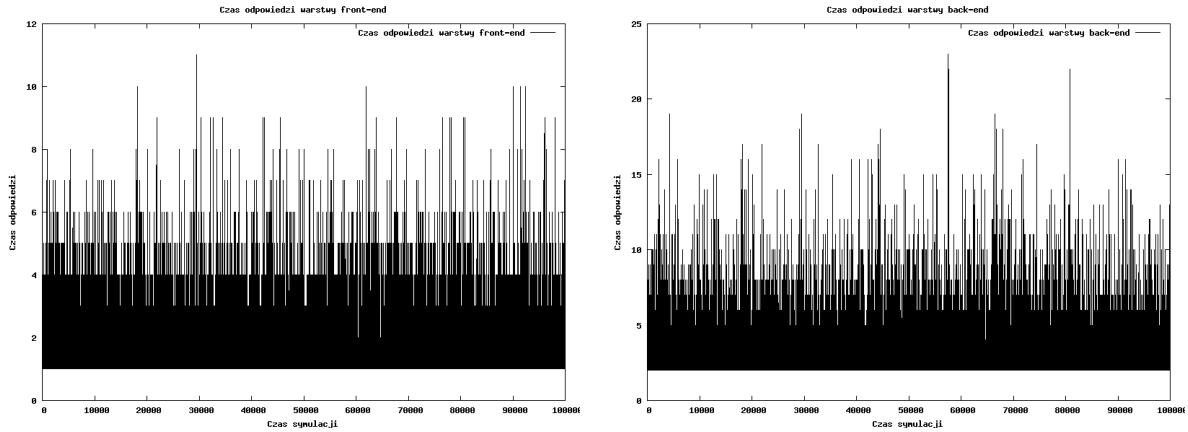



Rys. 4.8 „Strona” główna (środek rysunku) i „podstrony” (góra i dół rysunku) dla jednokierunkowego modelu TCPN klastrów obu warstw - model III (przykład)

Założono ponadto, że w modelowanym systemie prawdopodobieństwo rozdzielania zapytań pomiędzy poszczególne elementy w tej samej warstwie systemu wynosi *1/A* dla *front-end* i *1/B* dla *back-end*. Dla systemów kolejkowych zastosowano wykładniczą obsługę zapytań, bezstratność oraz nieograniczony czas oczekiwania w kolejce (kolejki: M/M/1/PS/$\infty$ i M/M/1/FIFO/$\infty$). Generowane w procesie wejścia znaczniki przekazywane są kolejno przez model klastra serwerów *front-end*, fragment sieci modelujący odrzucenie części zapytań oraz przez klaster bazy danych *back-end*. W prezentowanych modelach przyjęto, że systemy kolejkowe posiadają identyczne parametry.

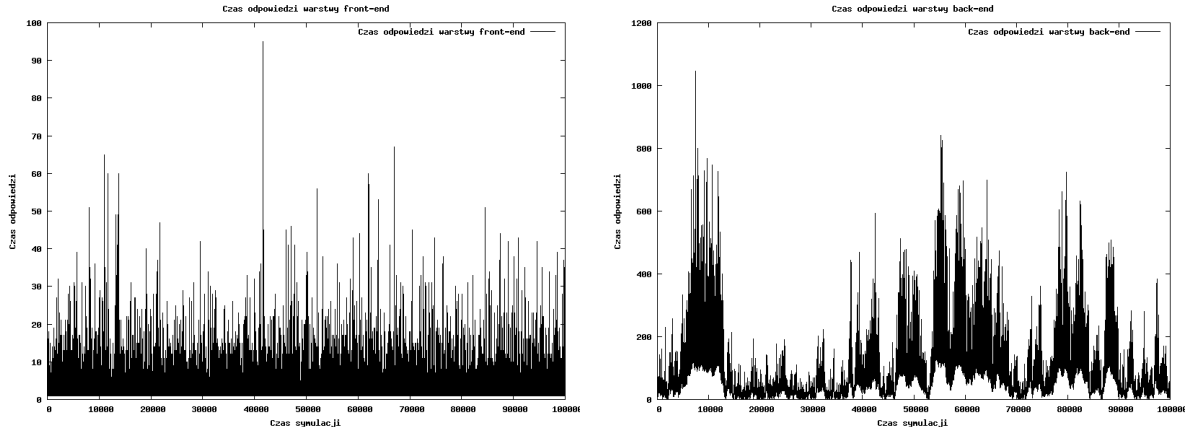
Na rysunkach 4.9, 4.10 i 4.11 zaprezentowano wyniki symulacji, dla czasów odpowiedzi w każdej z warstw, w Design/CPN dla prezentowanego przykładowego modelu (Rys. 4.8) przy założeniach, że: *A=2*, *B=2* (Podrozdz. 4.2) oraz dla:

- $\lambda = 1[1/s]$, $\mu = 10[1/s]$ (Rys. 4.9),
- $\lambda = 10[1/s]$, $\mu = 10[1/s]$ (Rys. 4.10),
- $\lambda = 100[1/s]$, $\mu = 10[1/s]$ (Rys. 4.11).



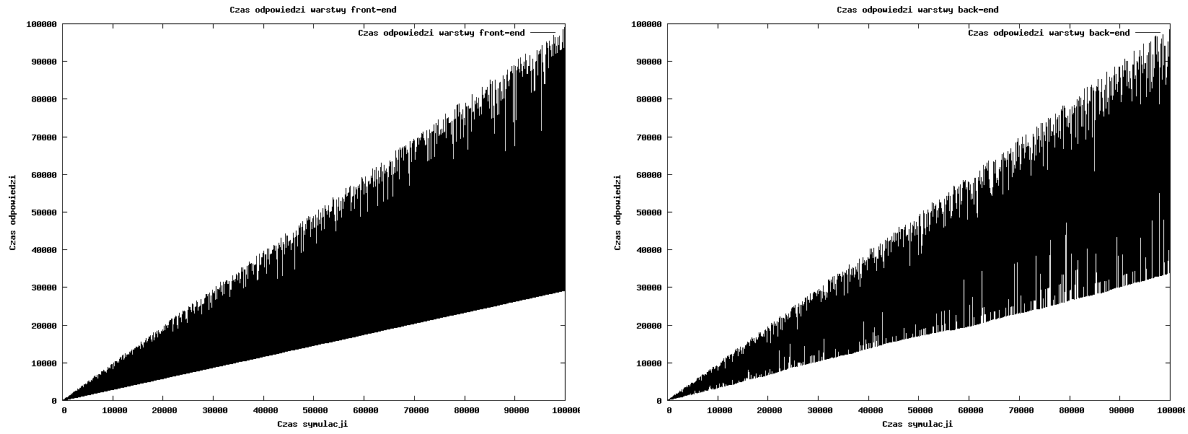

a) b)

Rys. 4.9 Czasy odpowiedzi [ms] dla $\lambda = 1[1/s]$ - jednokierunkowy model III: a) warstwy front-end i b) warstwy back-end




a) b)

Rys. 4.10 Czasy odpowiedzi [ms] dla $\lambda = 10[1/s]$ - jednokierunkowy model III: a) warstwy front-end i b) warstwy back-end

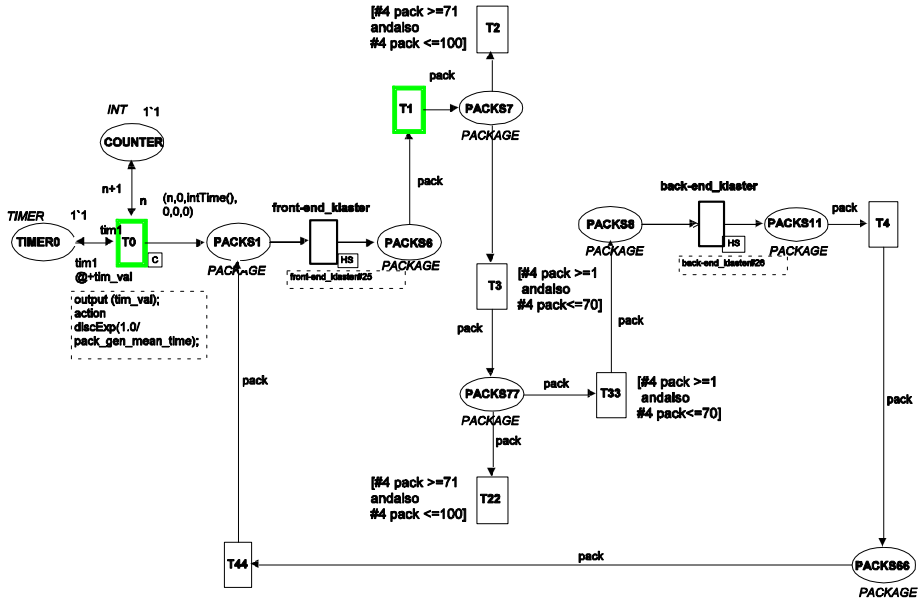



a) b)

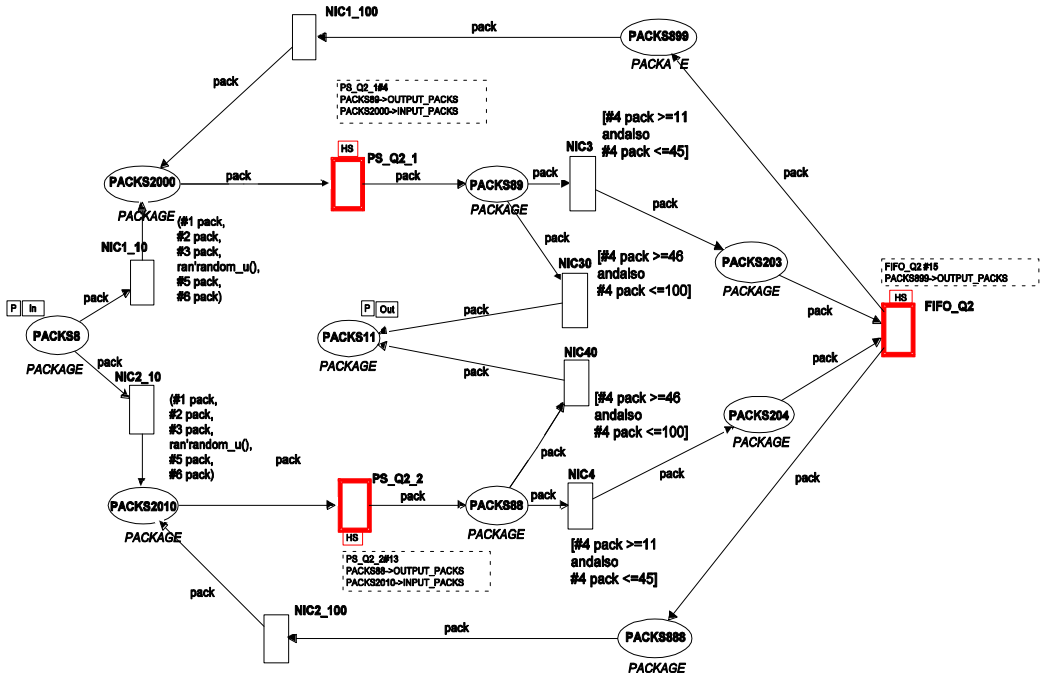
Rys. 4.11 Czasy odpowiedzi [ms] dla $\lambda = 100[1/s]$ - jednokierunkowy model III: a) warstwy front-end i b) warstwy back-end

Na rysunku 4.11 widać, że system nie jest w stanie nadążyć z obsługą napływających zapytań.

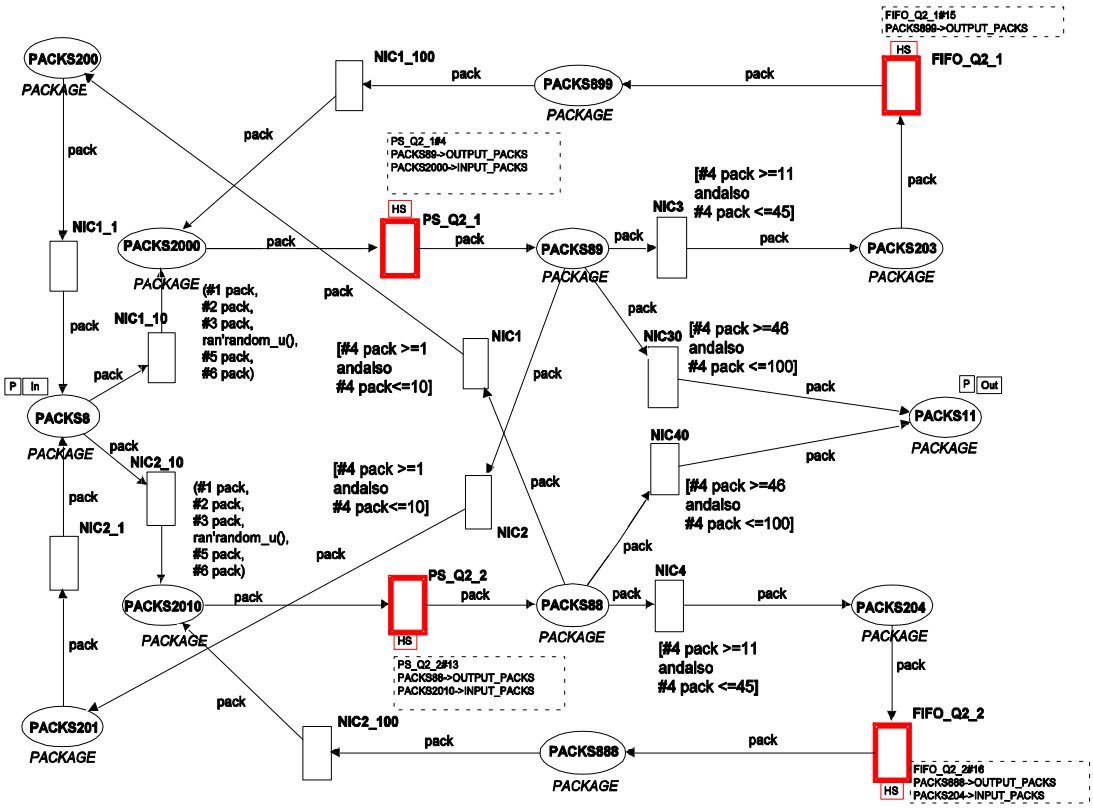
Model III z klastrami w obu warstwach różni się od jednokierunkowego modelu (Rys. 4.8) zarówno „stroną" główną modelu TCPN (Rys. 4.12), jak i „stroną" *back-end* (Rys. 4.13). „Strona" główna zawiera w tej wersji możliwość przekazywania znaczników z warstwy *back-end* ponownie do przetwarzania w warstwie *front-end*. Warstwą *back-end* modeluje możliwość wielokrotnego przekazywania znaczników do kolejki dysku serwera. Dla modelu III aktualna pozostaje górna część rysunku 4.8, modelująca warstwę *front-end*.




Rys. 4.12 „Strona" główna - modele III i IV (przykład)




Rys. 4.13 „Podstrona” z warstwą back-end - model III (przykład)




Rys. 4.14 „Podstrona” z warstwą back-end - model IV (przykład)

Ostatni prezentowany model IV (Rys. 4.6) dotyczy klastra w warstwie *front-end* i klastra z replikacją w warstwie *back-end*. „Strona" główna i „podstrona" *front-end* nie odbiegają od prezentowanej już dla modelu III (Rys. 4.12). Różnica w modelu TCPN, w porównaniu z modelem III, uwidacznia się dopiero w budowie „podstrony" z warstwą zawierającą replikację (*back-end*). „Podstronę" modelującą warstwę *back-end* dla replikacji bazy danych przedstawiono na rysunku 4.14. Pokazany przykład (Rys. 4.14) zawiera dwa węzły bazy danych (*B=2*) a jego najistotniejszymi własnościami są:

- możliwość kierowania znaczników do dowolnego węzła (lokalizacji replikacji) (*1/B*),
- powrót znaczników na początek warstwy,
- wykonanie synchronizacji danych w poszczególnych lokalizacjach (*pReplikacji*).

W przypadku modelu IV napotkano na problemy symulacji związane z przekazywaniem znaczników w procesie kolejnych wizyt w węzłach, co skutkowało coraz większym skomplikowaniem modelu systemu i wydłużaniem się czasu samej symulacji. Konieczność ciągłej modyfikacji, nawet graficzno-modułowej reprezentacji modelu, dla różnych złożonych architektur, jest czasochłonna. Natrafiono na [126] i prace [119, 120, 121], traktujące o sposobach modelowania różnych typów replikacji baz danych. Badania te bazowały na komercyjnym nieformalnym narzędziu CSIM, będącym środowiskiem programistycznym symulującym zdarzenia [25, 97].

### 4.2.3 MODELE CSIM

Poniżej przytoczony zostanie krótki opis budowy i działania modeli symulacyjnych CSIM umożliwiający zrozumienie sposobu modelowania, otrzymywania wyników oraz ich interpretacji. Zastosowanie pakietu CSIM, do modelownia parametrów wydajnościowych interaktywnych systemów internetowych, pociąga za sobą możliwość konfrontacji wyników z wynikami TCPN. Należy również wspomnieć o możliwościach CSIM rozszerzających liczbę parametrów systemów oraz łatwość skalowania modeli.

W podrozdziale, omówione zostanie zastosowanie wybranych bibliotek symulatora CSIM do modelowania ISIROSO. Bazując na modelach kolejkowych (Rys. 3.4, 4.4, 4.5 i 4.6) utworzono modele CSIM. Dzięki wykorzystaniu gotowych elementów CSIM, możliwe jest odwzorowanie modeli kolejkowych na modele symulacyjne.

Algorytm budowy programu symulacyjnego można opisać jako:

- proces symulacji - główną funkcją programu wywoływaną po jego uruchomieniu, jest funkcja `sim()`,
- tworzenie plików wynikowych (`fp = fopen(NAZWA_PL)`) - w pierwszej kolejności funkcja `sim()`otwiera pliki, w których zapisywane są wyniki symulacji,
- incjalizację obiektów związanych z warstwą *front-end* (`initA(LICZBA_WEZLOW_A)`) - wywoływana jest funkcja `initA()`, która inicjuje obiekty związane z symulacją, a w szczególności związane z warstwą *front-end*; na początku funkcja `initA` otwiera i ustawia plik, w którym będą zapisywane główne dane otrzymane z symulacji (funkcja `set_output_file(fp)`); ustawiana jest też nazwa symulowanego modelu `set_model_name()`; nazwa ta służy wyłącznie do generowania raportu;

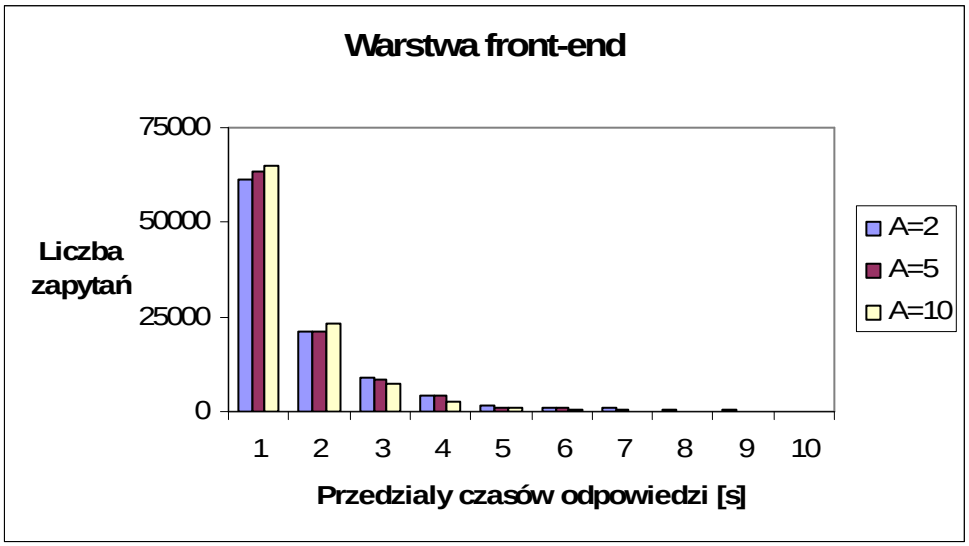
instrukcja `done = new event("done")` inicjuje obiekt `done`, czyli zdarzenie ustawiane po obsłużeniu ostatniego klienta,
    - tworzenie obiektu w pakiecie CSIM (`PS_Q1_A=new facility_set((NAZWA_PS_A,PS_Q1_A))` - inicjalizowana jest grupa obiektów `facility` odpowiadająca serwerom w warstwie *front-end*,
    - ustawienie sposobu szeregowania PS kolejki (`(*PS_Q1_A)[i].set_servicefunc(prc_shr))` - obsługa kolejki w każdym z serwerów,
    - tworzenie wskaźników do tabel zawierających raporty,
- incjalizację obiektów związanych z warstwą *back-end* (`initB(LICZBA_WEZLOW_B))` - wywołanie funkcji `initB()`, inicjalizującej obiekty związane z warstwą *back-end* [120]; są to dwa obiekty `facility` modelujące jednostki przetwarzające i dysk twardy,
    - tworzenie obiektu CSIM (`PS_Q2_B=new facility_set(NAZWA_PS_B,PS_Q2_B)`),
    - ustawienie sposobu szeregowania PS kolejki (`(*PS_Q2_B)[j].set_servicefunc(prc_shr)`),
    - tworzenie obiektu CSIM (`FIFO_Q2_B=new facility_set((NAZWA_FIFO_B,FIFO_Q2_B)`),
    - ustawienie sposobu szeregowania FIFO kolejki (`(*FIFO_Q2_B)[j].set_servicefunc(fcfs)`) - nie jest to konieczne, ze względu na domyślne ustawienie FIFO,
    - inicjalizowanie wskaźników do tabel, w których zapisywane będą raporty dotyczące warstwy *back-end*,
- tworzenie procesu głównego - wywołanie funkcji `create("sim")` sprawia, że główna funkcja staje się procesem w rozumieniu obiektów CSIM,
- wprowadzanie kolejnych klientów do kolejki oczekiwania na obsługę - funkcja `clientQueue(K)` powoduje dodanie klientów do kolejki oczekujących na obsługę. Funkcja ta na samym początku wykonuje instrukcje `create` przez co staje się obiektem typu `process`,
    - funkcja modelująca klientów (`create(K)`),
        - wywoływanie funkcji `client()`; jest to równoznaczne z przybyciem klienta i ponownym odczekaniem losowego odcinka czasu, za co z kolei odpowiada funkcja modelująca obsługę w warstwie,
        - modelowanie czasu generowania zapytania (`hold(exponential(CZAS_GEN_Z))`) - czyli czasu pomiędzy kolejnymi zapytaniami klienta,
- obsługa klienta w warstwie *front-end*,
    - modelowanie czasu obsługi w warstwie *front-end* (`(*PS_Q1_A)[i].use(exponential(CZAS_OBSLUGI))`),
- obsługa klienta w warstwie *back-end*,
    - modelowanie czasu obsługi jednostki przetwarzającej warstwy *back-end* (`(*PS_Q2_B)[j].use(exponential(CZAS_OBSLUGI))`),

  - modelowanie czasu obsługi w warstwie *back-end* (`(*FIFO_Q2_B)[j].use(exponential(CZAS_OBSLUGI))`),
  - w przypadku replikacji wywoływana jest funkcja replikacji `db_replication(TYP)`, z określonym typem, składająca się z funkcji omówionych w [120],
- zakończenie obsługi klienta,
- oczekiwanie na zakończenia czasu symulacji (`done->wait()`) - linia `done->wait()` powoduje, że cała symulacja trwa aż zostanie ustawione zdarzenie `done`,
- generowanie raportu i statystyk modelu (`report()`, `mdlstat()`) - funkcja `report()` generuje raport z przeprowadzonej symulacji; funkcja `mdlstat()` powoduje wygenerowanie statystyk modelu, są to m.in. czas użycia procesora, użycie pamięci, informacje o działających procesach itp.

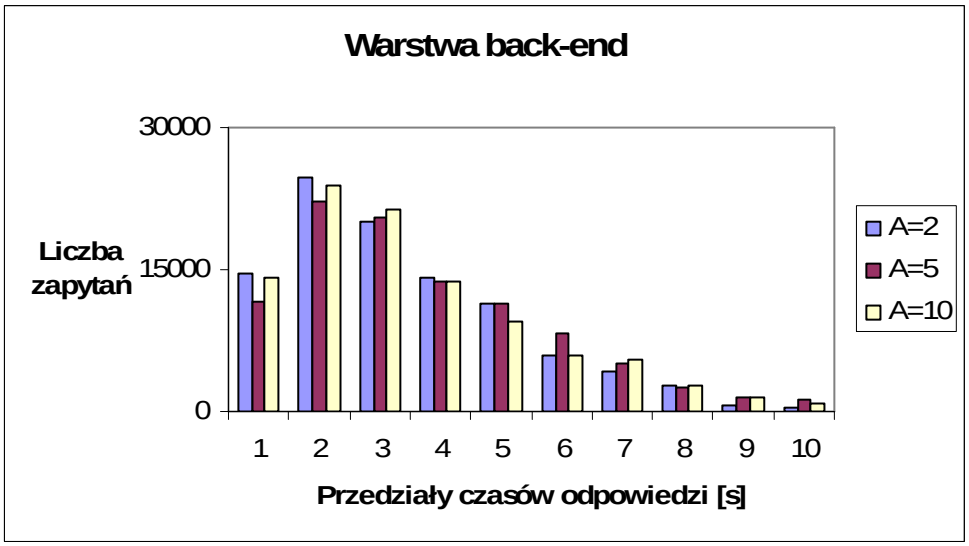
Funkcja `client()` modeluje obsługiwanego klienta. `model->note_entry()` powoduje wpisanie momentu rozpoczęcia obsługi klienta do tabeli zawierającej raport z całej symulacji modelu jak również tabel zawierających raporty z czasami obsługi w warstwach. Następnie generowana jest losowa liczba, która oznaczać będzie numer serwera warstwy *front-end*. W dalszej kolejności odnotowany zostaje w raporcie czas opuszczenia warstwy a obsługa przekazana jest do warstwy *back-end*, co wywołuje funkcję `db_processing()`. Jest to równoznaczne z zakończeniem obsługi zapytania w warstwie *front-end*. Jednocześnie ma miejsce sprawdzenie, czy nie upłynął czas symulacji, a jeżeli tak, ustawia się zdarzenie kończące symulacje `done->set()`. Funkcja `db_processing()` odnotowuje czas rozpoczęcia obsługi zapytania, losowo wybiera jeden z węzłów, zapytanie jest przetwarzane w wybranym serwerze przez losowy okres czasu. Na koniec odnotowywany jest czas opuszczenia warstwy *back-end*.

W CSIM istnieje wiele sposobów na generowanie liczb losowych o różnych rozkładach. Przykładowo funkcja `exponential()` służy do generowania liczb losowych o rozkładzie wykładniczym.

Wykresy (Rys. 4.15a,b) pokazują przykładowe wyniki symulacji modelu II klastra warstwy *front-end* (liczby zapytań z określonym czasem odpowiedzi) przy założeniach, że: *A={2,5,10}*, $\lambda = 300[1/s]$, $\mu = 50[1/s]$. Przy zwiększającej się liczbie węzłów przetwarzających warstwy *front-end*, zaobserwować można, że rośnie liczba zapytań o krótkich czasach realizacji dla warstwy *front-end*. W warstwie drugiej ze względu na większą liczbę zapytań trafiających do tej warstwy w krótszym czasie, w wyniku szybszej obsługi w warstwie pierwszej, czasy odpowiedzi ulegają nieznacznemu wzrostowi (Rys. 4.15b).



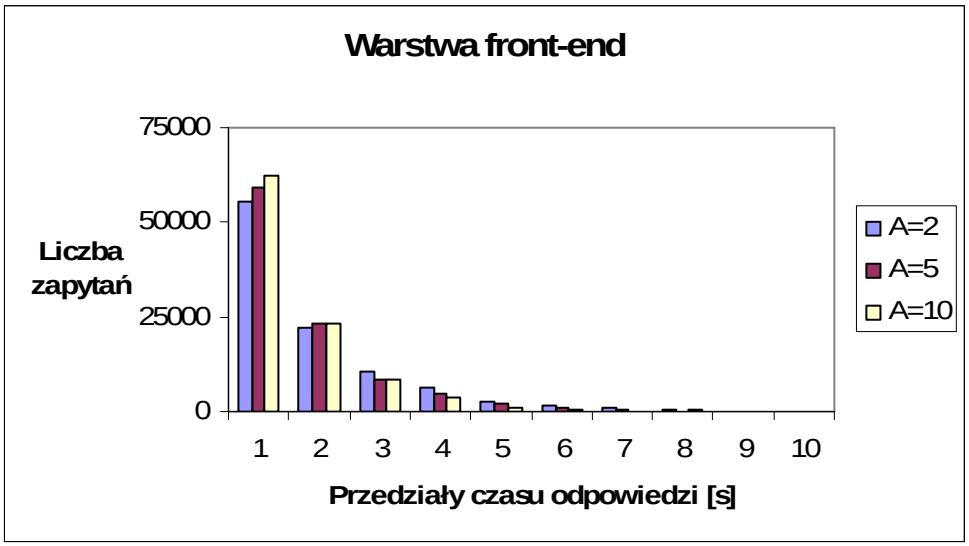

a)



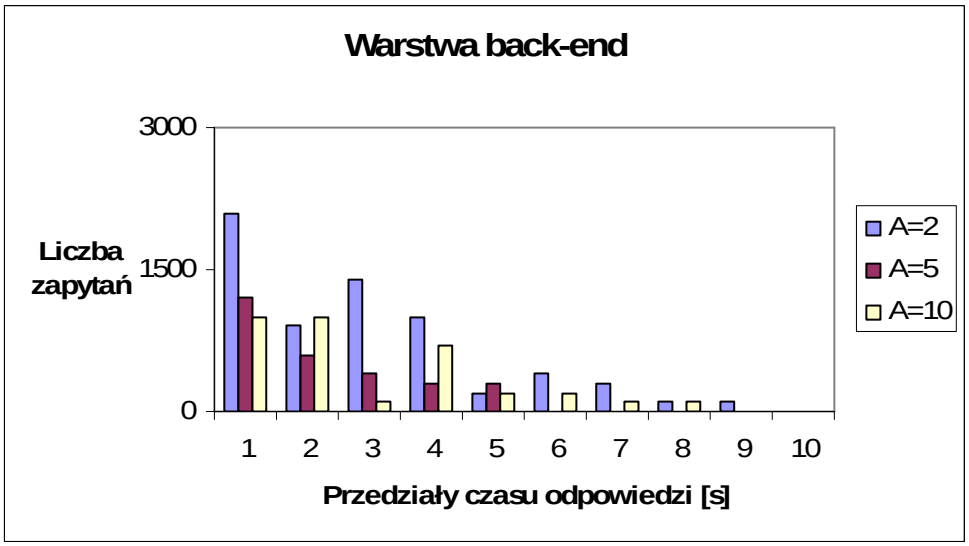

b)

Rys. 4.15 Rozkład liczby zapytań w poszczególnych przedziałach czasu odpowiedzi dla modelu klastra warstwy front-end (Rys. 4.4) - model II: a) warstwa front-end i b) warstwa back-end

Wykresy (Rys. 4.16a,b) zawierają przykładowe wyniki symulacji modelu III klastrów obu warstw (liczby zapytań z określonym czasem odpowiedzi) przy założeniach, że: *A={2,5,10}, B=2,* $\lambda = 300[1/s]$, $\mu = 50[1/s]$. Przy zwiększającej się liczbie węzłów przetwarzających warstwy *front-end*, rośnie odsetek zapytań o krótkich czasach odpowiedzi dla warstwy *front-end*, co oznacza, jak w poprzednim przypadku, przyspieszenie obsługi w tej warstwie. Natomiast w przypadku warstwy *back-end* w porównaniu do wyników modelu II z rysunku 4.15b zaobserwować można wyraźne opóźnienie i mniejszą liczbę zapytań o krótkich czasach odpowiedzi, co jest spowodowane większą liczbą napływających zapytań do obsługi w krótszym czasie.




a)




b)

Rys. 4.16 Rozkład liczby zapytań w poszczególnych przedziałach czasu odpowiedzi dla modelu klastrów front-end i back-end (Rys. 4.5) - model III: a) warstwa front-end i b) warstwa back-end

Pozostawiając proces przybywania i obsługi zapytań, a zmieniając liczbę elementów w poszczególnych warstwach, zaprezentowano w postaci tabel wyniki symulacji modelu II klastra warstwy *front-end* i modelu III klastrów obu warstw w tabeli 4.1.

Tab. 4.1 Model II dla A={2, 5, 10} i B=1 oraz model III dla A=2, B={2, 5, 10} - średni czas odpowiedzi [ms] (przykłady)

| A / Warstwa | 2 | 5 | 10 |
|---|---|---|---|
| front-end (model II) | 1040 | 996 | 995 |
| back-end (model II) | 2131 | 2131 | 2134 |
| **B / Warstwa** | **2** | **5** | **10** |
| front-end (model III) | 9982 | 9995 | 1101 |
| back-end (model III) | 12134 | 12079 | 12073 |

W przypadku modelu II klastra warstwy *front-end* wraz ze wzrostem liczby węzłów *front-end* skraca się czas odpowiedzi w tej warstwie oraz wydłuża w warstwie *back-end*. Na podstawie danych z tabeli 4.1 można zaobserwować nieznaczne wydłużenie czasu odpowiedzi dla *front-end* i skrócenie dla *back-end* w przypadku modelu III klastrów obu warstw przy zwiększającej się liczbie elementów warstwy *back-end*.

## 4.3 PODSUMOWANIE ROZDZIAŁU

Na wstępie rozdziału zaproponowano sposoby modyfikacji architektur systemów internetowych poprzez uwzględnienie dodatkowo klastrów i replikacji. Celem tej modyfikacji było sprostanie wymaganiom związanym z dużym obciążeniem systemów. Obok modelu podstawowego (model I), przedstawionego szczegółowo w rozdziale 3, opracowano modele ISIROSO zawierające klastry i replikację. Ze względu na różnice w budowie podzielono je na:

- klaster w warstwie *front-end* (Rys. 4.4) - model II,
- klaster w warstwie *front-end* i klaster w warstwie *back-end* (Rys. 4.5) - model III,
- klaster w warstwie *front-end* i klaster z replikacją w warstwie *back-end* (Rys. 4.6) - model IV.

Modele te zawierają proponowane modyfikacje architektury interaktywnego systemu szybkozmiennej giełdy internetowej z grupy ISIROSO.

Zastosowanie w systemach internetowych klastrów pociąga za sobą korzystne rozdzielanie zapytań. Rozkładanie ruchu dotyczy zapytań przybywających do warstwy *front-end*, ale również zleceń przesyłanych do warstwy *back-end*. Zastosowanie replikacji bazy danych w systemach internetowych pozwala na realizację niezależnych transakcji w różnych lokalizacjach. Do prezentacji zasad konstruowania architektur ISIROSO z klastrami i replikacją posłużono się modelami kolejkowymi. Do analizy zastosowano symulatory Design/CPN i CSIM. Owe narzędzia pozwalają na wyznaczanie charakterystyk parametrów istotnych ze względu na konieczną analizę wydajnościową systemu giełdy internetowej.

Niniejsze badania można traktować jako rozwinięcie doświadczeń [48] i własnych analiz [59, 64, 83, 84, 86, 87, 88, 89, 117].

# 5 ANALIZA ROZWIĄZAŃ

Celem rozdziału jest analiza parametrów wydajnościowych ISIROSO dla wybranych modeli architektur przedstawionych w rozdziałach 3 i 4. Są to modele warstwowe: podstawowy (model I), z klastrem *front-end* (model II), z klastrami *front-end* i *back-end* (model III) oraz z klastrem *front-end* i klastrem z replikacją *back-end* (model IV). Celem przeprowadzonej analizy modeli jest wykazanie ich przydatności do modelowania architektur systemów internetowych. Analiza bazuje na badaniach symulacyjnych TCPN i CSIM. Przedstawiono wybrane wyniki symulacji w celu zobrazowania zachowania systemu w różnych konfiguracjach. Zaprezentowanie charakterystyk z wszystkich wykonanych symulacji jest utrudnione ze względu na ich objętość. Wybrane charakterystyki zostały umieszczone w dodatku. Zdecydowano się na prezentację wybranych wyników w postaci wykresów oraz tabel.

Przyjęto następujący tok analizy rozwiązań. Do symulacji modeli z wykorzystaniem Design/CPN i CSIM użyto charakterystyk strumienia zgłoszeń o identycznych parametrach. Symulacje ograniczono do trzech wartości parametru rozkładu wykładniczego $\lambda = \{100[1/s],300[1/s],500[1/s]\}$, obrazującego trzy typy obciążenia niezależnie od konfiguracji systemów internetowych. Konfiguracje systemu internetowego ograniczono do:

- model I (*A*=1, *B*=1),
- model II (*A*=1, *B*=2), (*A*=1, *B*=4),
- model III (*A*=2, *B*=2), (*A*=4, *B*=2), (*A*=4, *B*=4),
- model IV (*A*=2, *B*=2), (*A*=4, *B*=2), (*A*=4, *B*=4),

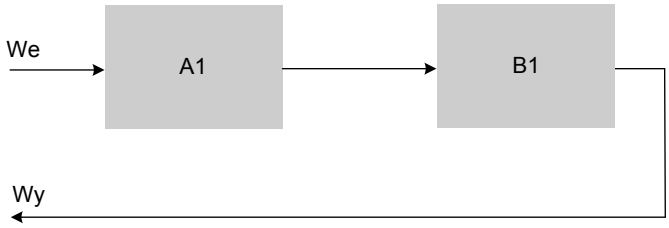
Wszystkie systemy obsługi, w modelowanym giełdowym systemie internetowym, posiadają, dla poszczególnych symulacji, tę samą wartość parametru $\mu = 100[1/s]$. Analizie poddawane są długości kolejek i czas odpowiedzi w poszczególnych warstwach modelowanego systemu w celu zobrazowania ich zachowania. Końcowym elementem analizy jest zestawienie wyników obu symulacji dla tych dwóch parametrów przy jednakowym czasie symulacji 100000[s].

## 5.1 SYMULACJE Z WYKORZYSTANIEM MODELI TCPN

Symulacje modeli TCPN bazują na module wydajnościowym narzędzia Design/CPN. Na początku analizie poddano długości kolejek w trakcie symulacji. W czasie trwania symulacji monitorowane są wszystkie kolejki modeli.

MODEL PODSTAWOWY (MODEL I)

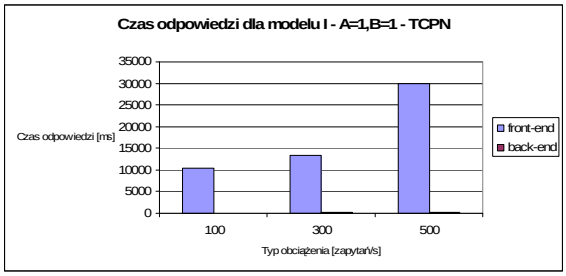
Analizowany model I (Rys. 5.1) w postaci TCPN przedstawiono na rysunku 3.8. Narastająca liczba zapytań powoduje wydłużenie kolejki warstwy *front-end* co skutkuje przedłużeniem czasu oczekiwania na odpowiedź obserwowanego w czasie symulacji. W warstwie *back-end* obserwuje się narastające długości kolejek PS i FIFO jednak nie są one tak krytyczne jak w przypadku *front-end*. Zwiększająca się liczba zapytań w systemie internetowym powoduje wydłużenie czasu odpowiedzi do poziomu, który nie może zostać zaakceptowany przez klienta. Dla modelu I w każdym przypadku obciążenia występuje niezrównoważenie systemu związane obserwowane jako wydłużenie się kolejki warstwy *front-end*.




Rys. 5.1 Model podstawowy *A*=1 i *B*=1

- Przypadek modelu I dla *A*=1, *B*=1 (Rys. 5.1) i $\lambda = 100[1/s]$ obrazuje (narastanie kolejki PS_Q*1_1* do nieskończoności) przeciążenie warstwy *front-end* ISIROSO. Skutkuje to niezrównoważeniem całego systemu mimo, że długości kolejek elementów warstwy *back-end* nie wykazują wzrostu. Przypadek modelu I dla *A*=1, *B*=1 i $\lambda = 300[1/s]$ obrazuje, poprzez dalsze narastanie kolejki PS_Q*1_1* do nieskończoności, przeciążenie warstwy *front-end* ISIROSO. Skutkuje to, podobnie jak w poprzednim przypadku, niezrównoważeniem całego systemu. Tym razem długości kolejek elementów warstwy *back-end* wynoszą średnio około 1,5 dla PS_Q*2_1* i około 1,3 dla FIFO_Q*2*. Wykazują one wzrost, ale nie jest on krytyczny dla zrównoważenia systemu. Przypadek modelu I dla *A*=1, *B*=1 i $\lambda = 500[1/s]$ obrazuje, poprzez dalsze narastanie kolejki PS_Q*1_1* do nieskończoności, coraz większe przeciążenie warstwy *front-end* modelu ISIROSO. Skutkuje to, podobnie jak w poprzednich przypadkach, niezrównoważeniem całego systemu. Tym razem długości kolejek elementów warstwy *back-end* wynoszą średnio około 6,9 dla PS_Q*2* i około 6 dla FIFO_Q*2*. Wykazują one wzrost, ale podobnie jak dla przypadku $\lambda = 300[1/s]$ nie jest to spowodowane niezrównoważeniem systemu.

Na wykresie (Rys. 5.2) zaprezentowano czasy odpowiedzi poszczególnych warstw systemu dla modelu I przy trzech różnych typach obciążenia $\lambda = \{100[1/s], 300[1/s], 500[1/s]\}$. Wykres (Rys. 5.2) ilustruje omawiane niezrównoważenie systemu w warstwie *front-end*.

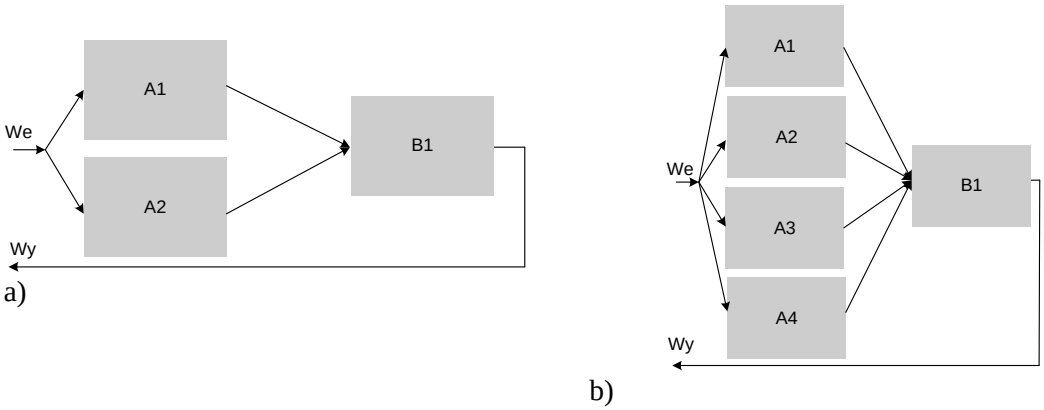



Rys. 5.2 Czasy odpowiedzi warstw (front-end, back-end) modelu I dla trzech typów obciążeń

MODEL KLASTRA FRONT-END (MODEL II)

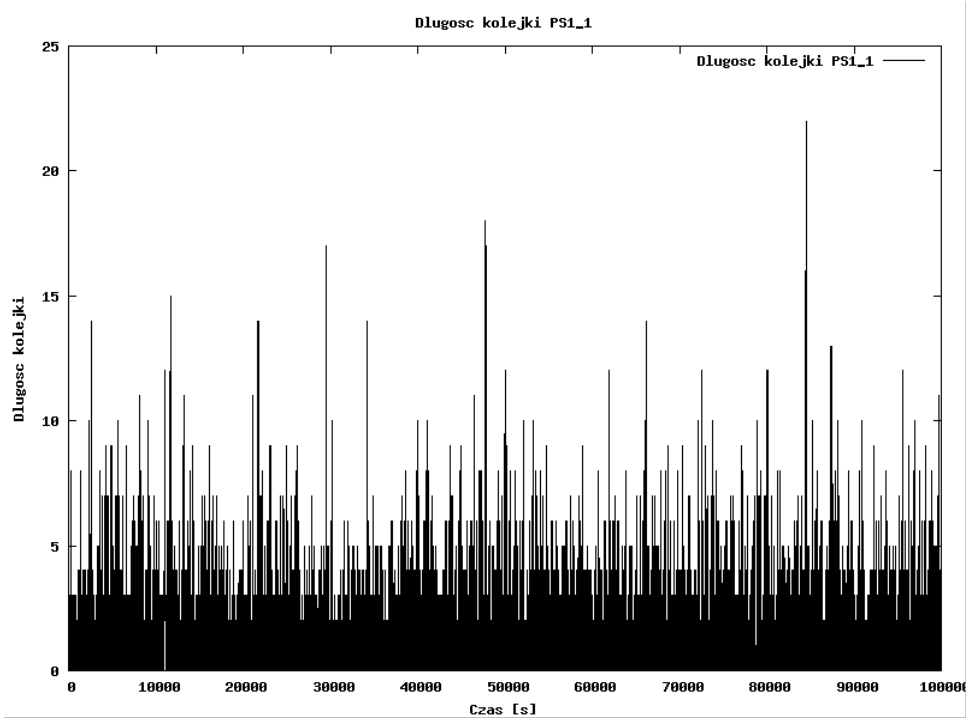
Symulowany model II (Rys. 5.3) przedstawiono w postaci TCPN w podrozdz. 4.2.2. Analiza długości kolejek, w przypadku, gdy w warstwie *front-end* zainstalowano klaster 2 i 4 węzłowy (wprowadzono tym samym równoległą obsługę zapytań), dla przypadku 100[zapytań/s] pozwalałaby sądzić, że problem został rozwiązany. Jednak już w przypadku

300[zapytań/s] kolejki warstwy *front-end* wydłużają się. Podobnie dzieje się z kolejką PS_Q*2_1*, gdzie długość jej wzrosła prawie dziesięciokrotnie. Dla 500[zapytań/s] zrównoważenie systemu zostaje całkowicie zachwiane. Liczba zapytań w kolejce FIFO szczególnie w przypadku 500[zapytań/s] zaczyna niepokojąco rosnąc. Całościowo z narastającą liczbą zapytań nieco lepiej radzi sobie klaster 4 węzłowy w warstwie *front-end*, ale i on dla przypadku 500[zapytań/s] wydłuża czas oczekiwania na odpowiedź. W przypadku modelu klastra w warstwie *front-end* niezrównoważenie systemu występuje dla 500[zapytań/s] zarówno dla klastra *A*=2 (Rys. 5.3a) i *A*=4 (Rys. 5.3b). Powoduje to z wydłużaniem wszystkich kolejek PS zarówno w warstwie klastra jaki warstwie *back-end*.

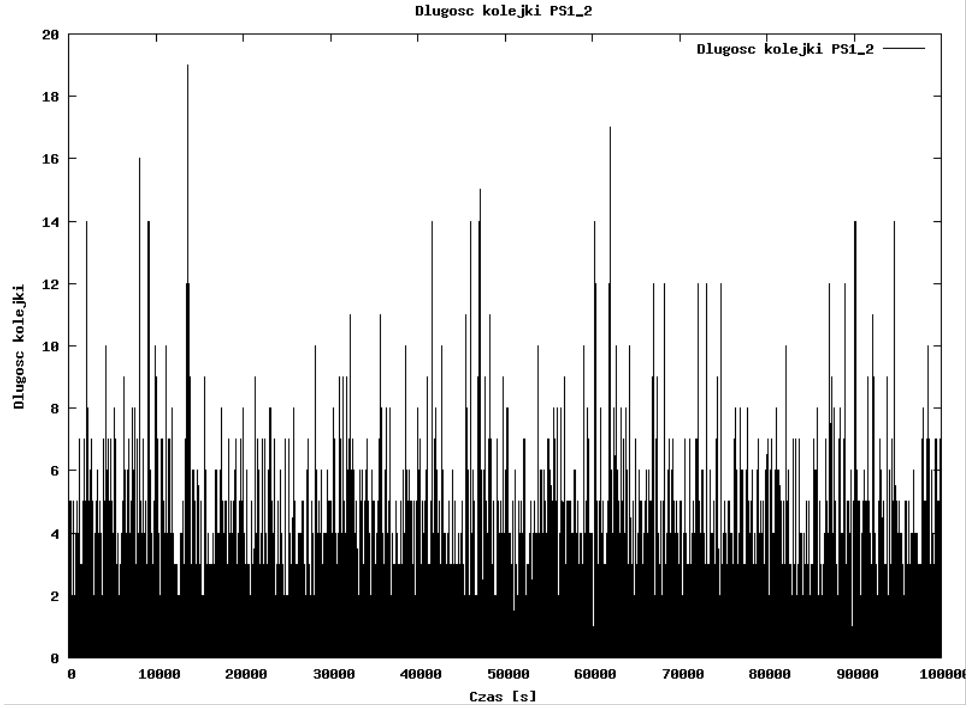



Rys. 5.3 Model klastra front-end (przykłady): a) *A*=2 i *B*=1, b) *A*=4 i *B*=1

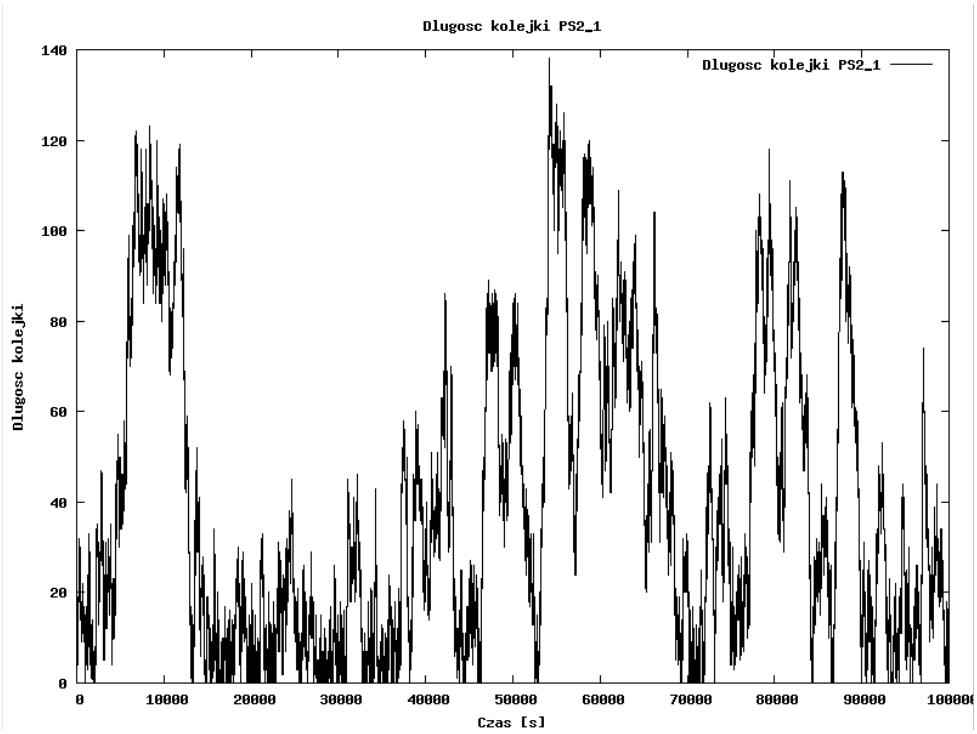
- Charakterystyki modelu II dla *A*=2, *B*=1 i $\lambda = 100[1/s]$ wskazują na zrównoważenie systemu. Średnie długości kolejek nie przekraczają wartości kilkunastu zapytań oczekujących. Na rysunku 5.4 zaprezentowano charakterystyki obciążeń modelu II dla *A*=2, *B*=1 i $\lambda = 300[1/s]$. Charakterystyki tego modelu prezentują pogorszenie zrównoważenia systemu. Uwidacznia się to szczególnie w przypadku długości kolejki PS_Q*2_1* (Rys. 5.4c). Nie wskazuje to jednak na niezrównoważenie systemu. Zwiększenie obciążenia nie pociągnęło za sobą utraty zrównoważenia. Charakterystyki warstwy *front-end* (Rys. 5.4a,b) prezentują zrównoważenie przy średniej długości około kilku zapytań w kolejce. Podobnie zachowuje się kolejka FIFO_Q*2_1* (Rys. 5.4d). Charakterystyki modelu II dla *A*=2, *B*=1 i $\lambda = 500[1/s]$ obrazują utratę zrównoważenia systemu. Wszystkie kolejki narastają do nieskończoności. Jedynie wartość kolejki FIFO warstwy *back-end* jest mniejsza. Kolejne zwiększenie obciążenia skutkuje utratą zrównoważenia.



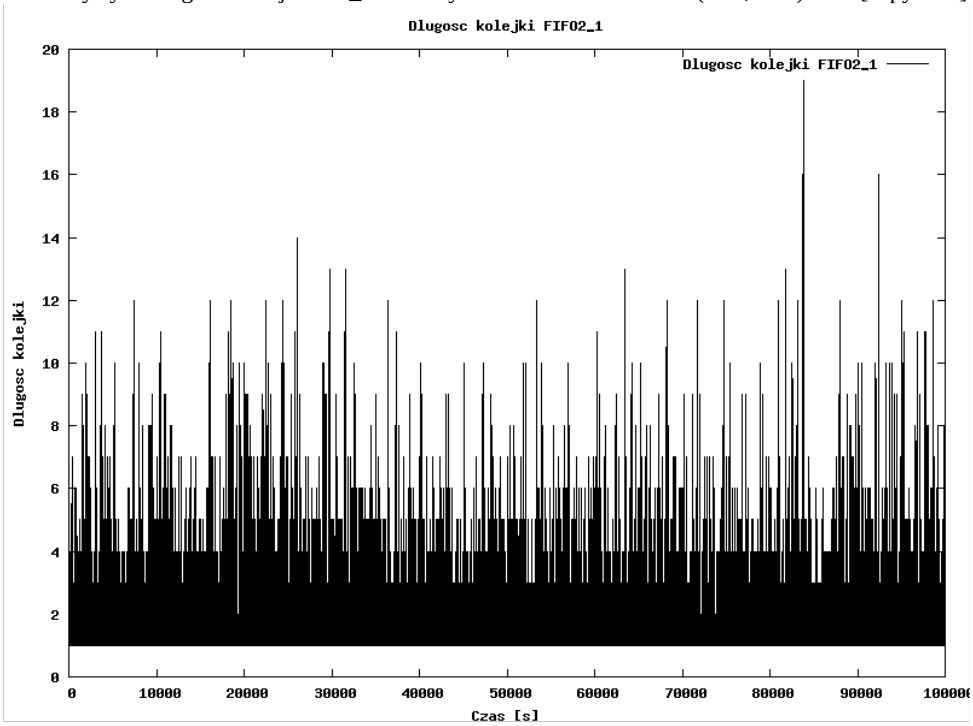

a) Charakterystyka długości kolejki PS1_1 warstwy front-end dla modelu II (*A*=2, *B*=1) - 300[zapytań/s]




b) Charakterystyka długości kolejki PS1_2 warstwy front-end dla modelu II (*A*=2, *B*=1) - 300[zapytań/s]




c) Charakterystyka długości kolejki PS2_1 warstwy back-end dla modelu II (*A*=2, *B*=1) - 300[zapytań/s]

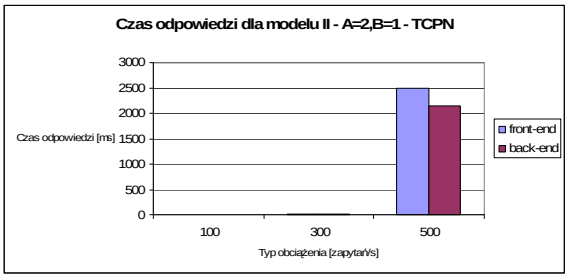



d) Charakterystyka długości kolejki FIFO2_1 warstwy back-end dla modelu II (*A*=2, *B*=1) - 300[zapytań/s]

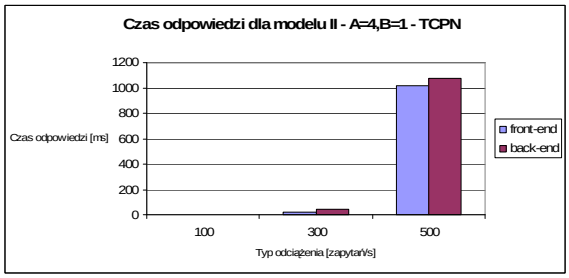
Rys. 5.4 Charakterystyki długości kolejek dla modelu II klastra 2 węzłowego front-end (300[zapytań/s])

- Charakterystyki modelu II dla *A*=4, *B*=1 i $\lambda = 100[1/s]$, jak się można spodziewać po analizie przypadku identycznego modelu dla *A*=2, obrazują zrównoważenie systemu. Liczba zapytań oczekujących w kolejkach nie przekracza w tym przypadku 10. Charakterystyki modelu II dla *A*=4, *B*=1 i $\lambda = 300[1/s]$, prezentują niezrównoważenie systemu mimo zwiększenia liczby serwerów warstwy *front-end*. Jest to sytuacja podobna od tej jaka była w identycznym modelu dla *A*=2. Zwiększenie liczby elementów klastra doprowadziło do skrócenia długości kolejki PS_Q*2_1*. Charakterystyki modelu II dla *A*=4, *B*=1 i $\lambda = 500[1/s]$, prezentują niezrównoważenie systemu. Wyraźna poprawa widoczna jest poprzez zmniejszenie długości kolejek, dzięki zwiększeniu liczby elementów klastra, w stosunku do identycznego modelu dla przypadku *A*=2, ale niezrównoważenie pozostaje.

Na wykresach (Rys. 5.5) zaprezentowano czasy odpowiedzi poszczególnych warstw systemu dla modelu II przy trzech różnych typach obciążenia $\lambda = \{100[1/s], 300[1/s], 500[1/s]\}$. Wykres (Rys. 5.5a) prezentuje przypadek *A*=2, natomiast wykres (Rys. 5.5b) prezentuje przypadek *A*=4. Zaobserwować można skrócenie czasu odpowiedzi przy największym stosowanym obciążeniu.




a)



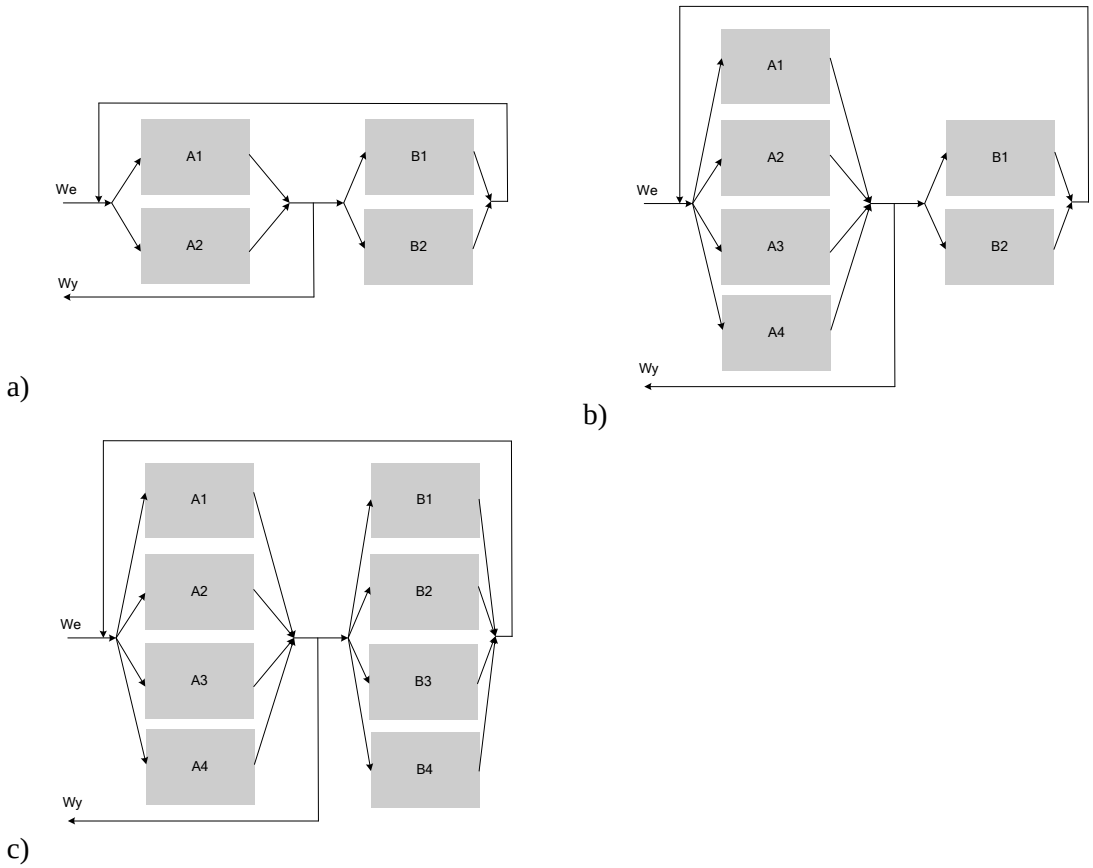

b)

Rys. 5.5 Czasy odpowiedzi warstw (front-end, back-end) przy trzech typach obciążeń dla przykładów modelu II: a) A=2, B=1, b) A=4, B=1

MODEL KLASTRÓW OBU WARSTW (MODEL III)

Kolejnym analizowanym modelem jest model z klastrami w obu warstwach (Rys. 5.6). Model III w postaci TCPN zaprezentowano na rysunkach 4.12 i 4.13. Widać poprawę wydajności w przypadku klastra *front-end*, ale dotyczy ona jedynie warstwy z klastrem w przypadku $\lambda = 500[1/s]$. Dla *A*=2 i *B*=2 długości kolejek PS_Q*1_1* i PS_Q*1_2* rosną wraz ze wzrostem liczby zapytań. W przypadku *A*=4 i *B*=2 długości kolejek PS_Q*1_1*, PS_Q*1_2*, PS_Q*1_3* i PS_Q*1_4* kształtują się na podobnym poziomie dla 100 i 300[zapytań/s] i rosną prawie 100 razy dla 500[zapytań/s]. W przypadku *A*=4 i *B*=4 długości kolejek PS_Q*1_1*, PS_Q*1_2*, PS_Q*1_3* i PS_Q*1_4* rosną stopniowo kilka razy dla każdego z trzech poziomów obciążeń. Wzrost ten dla 500[zapytań/s] powoduje niezrównoważenie w warstwie *front-end*. Obciążenie zapytaniami kolejek PS_Q*2_1* i PS_Q*2_2* dla przypadku *B*=2, PS_Q*2_1*, PS_Q*2_2*, PS_Q*2_3* i PS_Q*2_4* oraz przypadku *B*=4 utrzymuje się na jednakowym poziomie wskazując za każdym razem na niezrównoważenie systemu w warstwie *back-end*. Dla przypadku *B*=4 liczba zapytań w tych kolejkach była około dwukrotnie mniejsza. W kolejce FIFO liczba zapytań

w żadnym przypadku obciążenia nie przekroczyła 8. W wyniku oczekiwania na realizację zapytań w kolejkach PS_Q2_*B* czas odpowiedzi systemu nadal wydłuża się. Zastosowanie podobnej liczby węzłów w obu warstwach, w przypadku 4 węzłów, daje efekt w postaci skrócenia długości wszystkich kolejek. W tym przypadku niezrównoważenie systemu nie jest tak wyraźne dla 100[zapytań/s]. Niezrównoważenie systemu występuje we wszystkich pozostałych przypadkach.

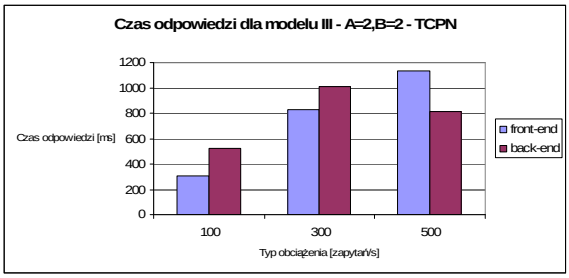



Rys. 5.6 Model klastrów obu warstw (przykłady): a) *A*=2 i *B*=2, b) *A*=4 i *B*=2, c) *A*=4 i *B*=4

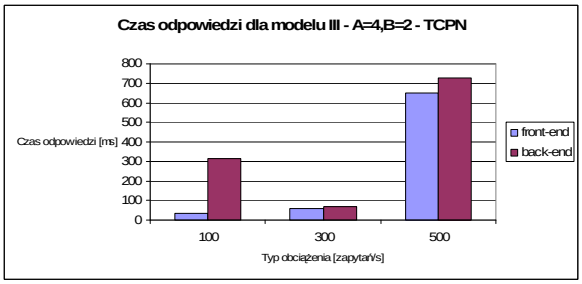
- Charakterystyki modelu III dla *A*=2, *B*=2 (Rys. 5.6a) i $\lambda = 100[1/s]$, przedstawiają niezrównoważenie systemu. Niezrównoważenie pokazują szczególnie kolejki PS warstwy *back-end*. Charakterystyki modelu III dla *A*=2, *B*=2 i $\lambda = \{300[1/s], 500[1/s]\}$, podobnie jak w poprzednim przypadku, przedstawiają niezrównoważenie systemu. Tym razem niezrównoważenie prezentują kolejki PS obu warstw.
- Charakterystyki modelu III dla *A*=4, *B*=2 (Rys. 5.6b) i $\lambda = \{100[1/s], 300[1/s]\}$, przedstawiają niezrównoważenie systemu. Niezrównoważenie obrazują kolejki PS warstwy *back-end*. Charakterystyki modelu III dla *A*=4, *B*=2 i $\lambda = 500[1/s]$, przedstawiają również niezrównoważenie systemu. Niezrównoważenie pokazują kolejki PS obu warstw.

- Charakterystyki modelu III dla *A*=4, *B*=4 (Rys. 5.6c) i $\lambda = 100[1/s]$, przedstawiają niezrównoważenie systemu. Niezrównoważenie prezentują kolejki PS warstwy *back-end*, jednak można zaobserwować prawie dziesięciokrotne ich zmniejszenie w porównaniu do identycznego modelu dla *B*=2. Charakterystyki modelu III dla *A*=4, *B*=4 i $\lambda = 300[1/s]$, przedstawiają niezrównoważenie systemu. Niezrównoważenie prezentują kolejki PS warstwy *back-end*, jednak można zaobserwować, tak jak w przypadku mniejszego obciążenia, choć już nie tak wyraźne, zmniejszenie długości kolejek w porównaniu do identycznego modelu dla *B*=2. Charakterystyki modelu III dla *A*=4, *B*=4 i $\lambda = 500[1/s]$, przedstawiają niezrównoważenie systemu. Niezrównoważenie obrazują kolejki PS obu warstw, jednak nie w takim stopniu jak dla przypadku tego samego modelu dla identycznych parametrów za wyjątkiem *B*=2. Na tej podstawie można wnioskować o celowości stosowania podobnej liczby węzłów w obu warstwach.

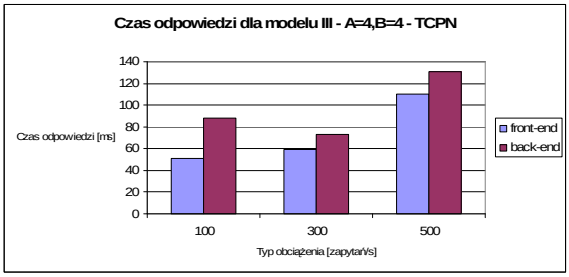
Na wykresach (Rys. 5.7) zaprezentowano czasy odpowiedzi poszczególnych warstw systemu dla modelu III przy trzech różnych typach obciążenia $\lambda = \{100[1/s], 300[1/s], 500[1/s]\}$. Wykres (Rys. 5.7a) prezentuje przypadek *A*=2, *B*=2, wykres (Rys. 5.7b) prezentuje przypadek *A*=4, *B*=2, natomiast wykres (Rys. 5.7c) prezentuje przypadek *A*=4, *B*=4. Zaobserwować można wydłużenie czasu odpowiedzi obu warstw przy największym stosowanym obciążeniu. Dla przypadków (Rys. 5.7a,b) i $\lambda = 500[1/s]$ czas odpowiedzi warstwy *front-end* powoduje niezrównoważenie systemu. W przypadku największego stosowanego obciążenia dla modelu (Rys. 5.7c) kolejki prezentują niezrównoważenie, jednak wyraźnie mniejsze.




a)




b)




c)

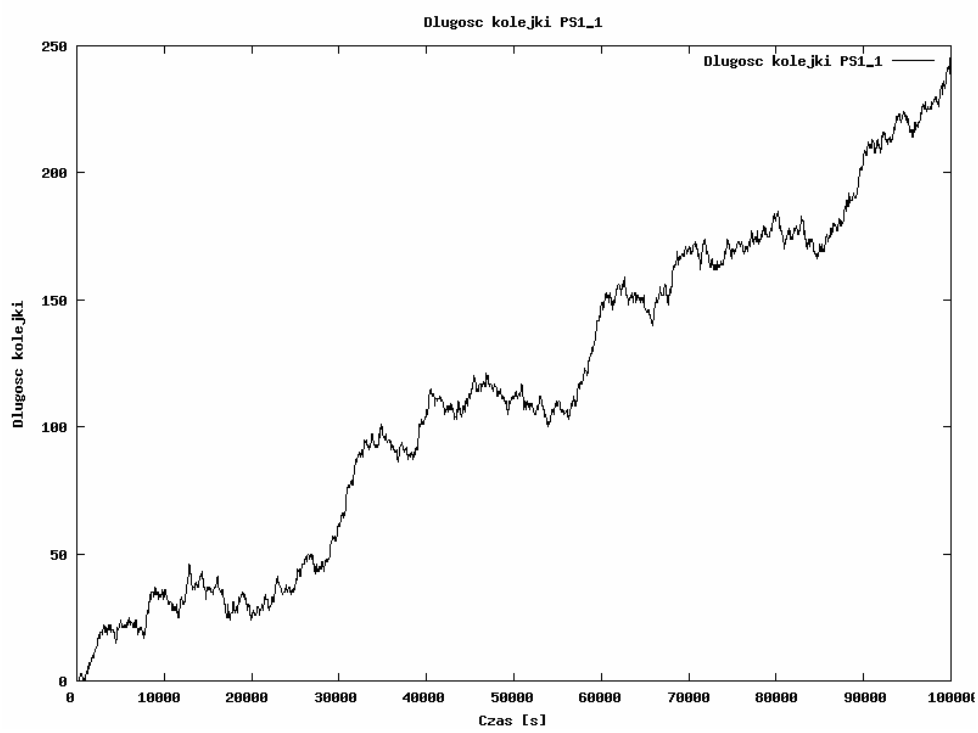
Rys. 5.7 Czasy odpowiedzi warstw (front-end, back-end) przy trzech typach obciążeń dla przykładów modelu III: a) A=2, B=2, b) A=4, B=2, c) A=4, B=4

MODEL WARSTWOWY KLASTRA I REPLIKACJI (MODEL IV)

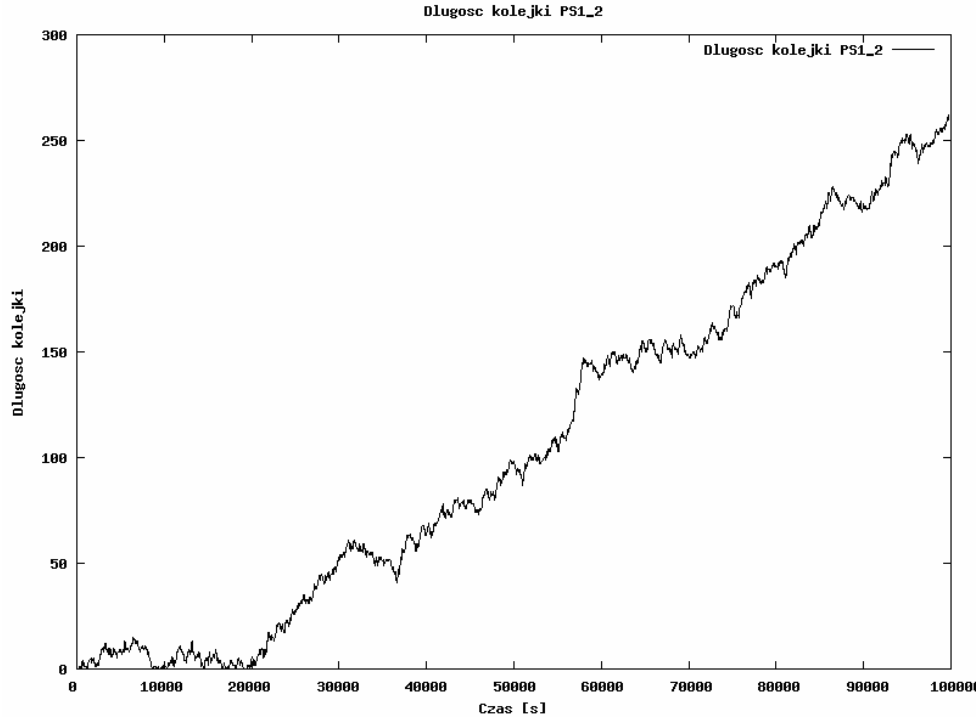
Model IV w postaci TCPN znaleźć można na rysunkach 4.12 i 4.14. Dla modelu z replikacją w warstwie *back-end* dopiero w przypadku 4 węzłów *front-end* i 4 węzłów *back-end* następuje zrównoważenie długości wszystkich kolejek. W przypadku *A*=2 i *B*=2 długość kolejek PS_Q*1_1* i PS_Q*1_2* rośnie wraz ze wzrostem liczby zapytań przedstawiając niezrównoważenie systemu w warstwie *front-end*. Podobnie w przypadku *A*=4 i *B*=2, *A*=4 i *B*=4 długość kolejek PS_Q*1_1*, PS_Q*1_2*, PS_Q*1_3* i PS_Q*1_4* rosną. Jednak w przypadku *A*=4 i *B*=4 wzrost liczby zapytań oczekujących w kolejce nie jest tak już taki duży. Liczba zapytań w kolejkach PS_Q*2_B* pozostaje na poziomie modeli klastra obu warstw. W przypadku kolejek FIFO liczba zapytań w żadnym przypadku obciążenia nie przekroczyła 5. Analiza liczby zapytań w kolejkach mówi o ciągle niedostatecznym poziomie obsługi zapytań, ale w przypadku *A*=4 i *B*=4 dla 500[zapytań/s] wzrost w porównaniu 300[zapytań/s] nie jest tak duży.

W przypadku modelu (model IV) klastra warstwy *front-end* i replikacji warstwy *back-end* zaobserwować można odwrotną sytuację jak w poprzednich przypadkach. Tutaj powodem największego niezrównoważenia systemu są kolejki PS drugiej warstwy. Jednak wraz ze wzrostem liczby węzłów przetwarzających w warstwie *back-end*, mimo zwiększającego się obciążenia zewnętrznego, system internetowy jest zrównoważony, ściślej ujmując nie popada w skrajne przypadki niezrównoważenia. Zachowanie to jest podobne do modelu III jednak w tym przypadku zaobserwować można zmniejszenie liczby i wyrównanie rozłożenia zapytań w kolejkach.

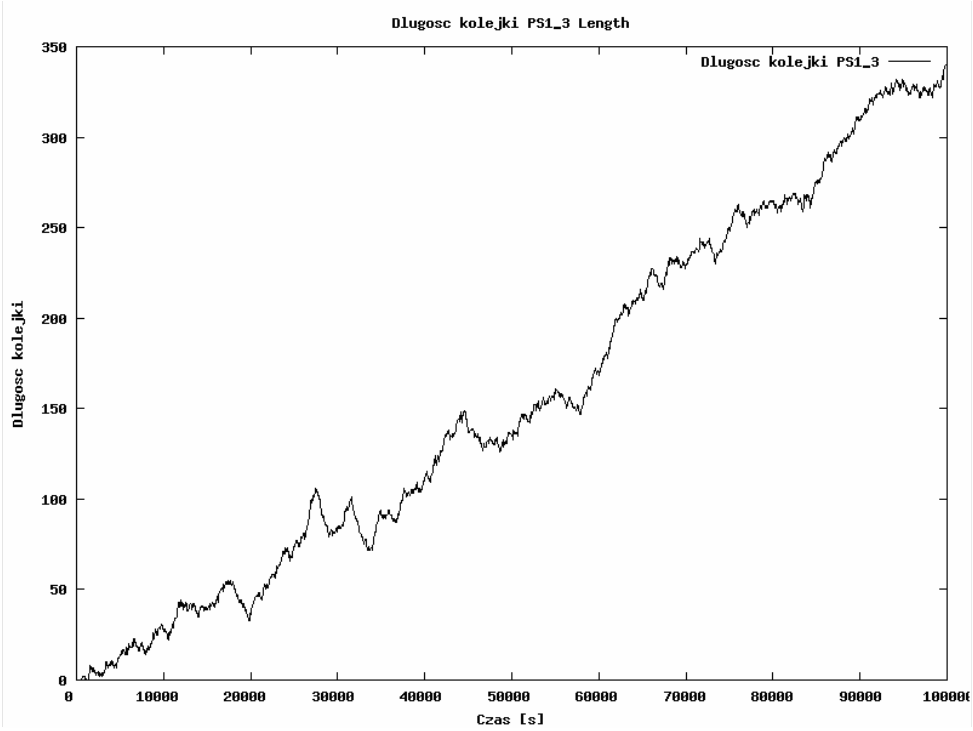
- Charakterystyki modelu IV dla *A*=2, *B*=2 i $\lambda = 100[1/s]$, przedstawiają niezrównoważenie systemu. Prezentują to kolejki PS warstwy *back-end*. Charakterystyki modelu IV dla *A*=2, *B*=2 i $\lambda = \{300[1/s],500[1/s]\}$, przedstawiają dalsze pogłębienie niezrównoważenia systemu. Tym razem niezrównoważenie prezentują kolejki PS obu warstw.
- Charakterystyki modelu IV dla *A*=4, *B*=2 i $\lambda = \{100[1/s],300[1/s]\}$, przedstawiają niezrównoważenie systemu. Obrazują to kolejki PS warstwy *back-end*. Charakterystyki modelu IV dla *A*=4, *B*=2 i $\lambda = 500[1/s]$, przedstawiają niezrównoważenie systemu. W tym przypadku niezrównoważenie prezentują kolejki PS obu warstw.
- Charakterystyki modelu IV dla *A*=4, *B*=4 i $\lambda = 100[1/s]$, przedstawiają niezrównoważenie systemu. Pokazują to kolejki PS warstwy *back-end*, ale nie jest ona tak wyraźna jak w poprzednich przypadkach dla modelu IV. Podobne są natomiast do charakterystyk modelu III dla *A*=4, *B*=4 i $\lambda = 100[1/s]$. Charakterystyki modelu IV dla *A*=4, *B*=4 i $\lambda = 300[1/s]$, przedstawiają niezrównoważenie systemu. Obrazują to kolejki PS warstwy *back-end*, jednak, i w tym przypadku, skróceniu uległy kolejki warstwy *back-end* w porównaniu z odpowiadającym temu modelowi, modelem III. Na rysunku 5.2 zaprezentowano charakterystyki modelu IV dla *A*=4, *B*=4 i $\lambda = 500[1/s]$. Charakterystyki modelu IV dla *A*=4, *B*=4 i $\lambda = 500[1/s]$, przedstawiają dążenie do zrównoważenia systemu. Długość kolejek PS warstwy *back-end* (Rys. 5.7e,g,i,k) jest dwukrotnie mniejsza niż dla przypadku *A*=4, *B*=4 i $\lambda = 500[1/s]$ modelu III. Dzieje się tak zarówno dla kolejek PS warstwy *front-end* (Rys. 5.7a,b,c,d) jak i FIFO warstwy *back-end* (Rys. 5.7f,h,j,l).



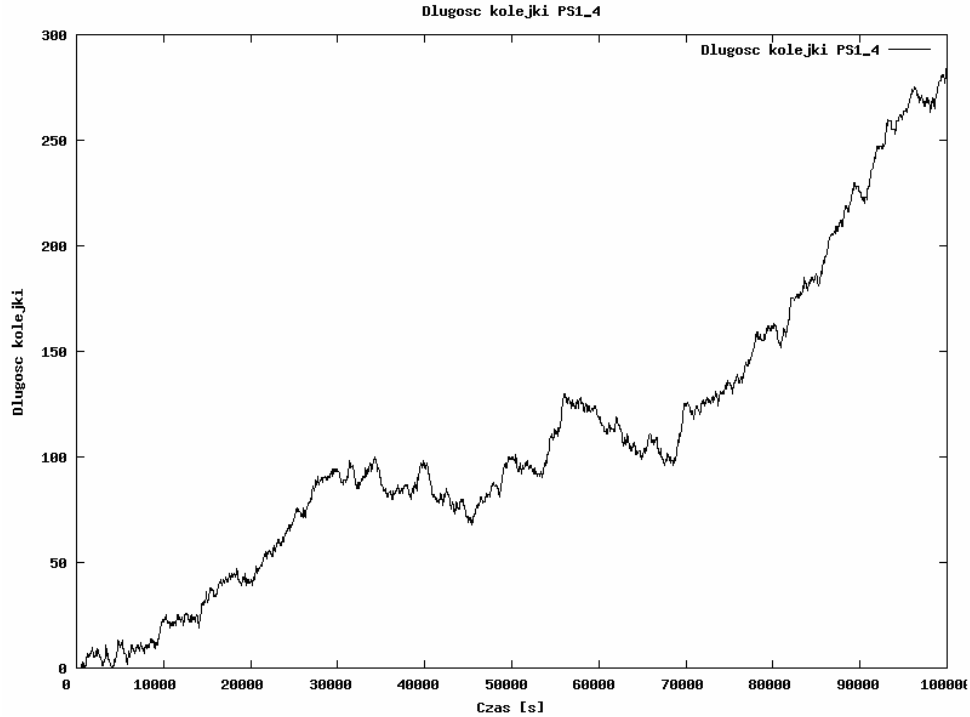

a) Charakterystyka długości kolejki PS1_1 warstwy front-end dla modelu IV (*A*=4, *B*=4) - 500[zapytań/s]



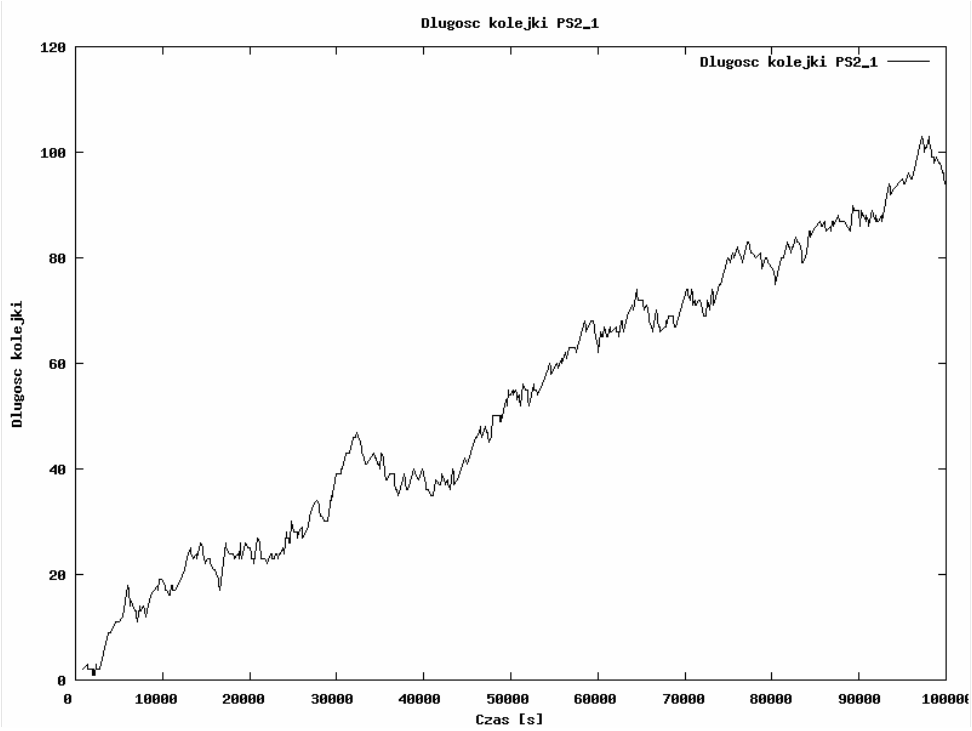

b) Charakterystyka długości kolejki PS1_2 warstwy front-end dla modelu IV (*A*=4, *B*=4) - 500[zapytań/s]



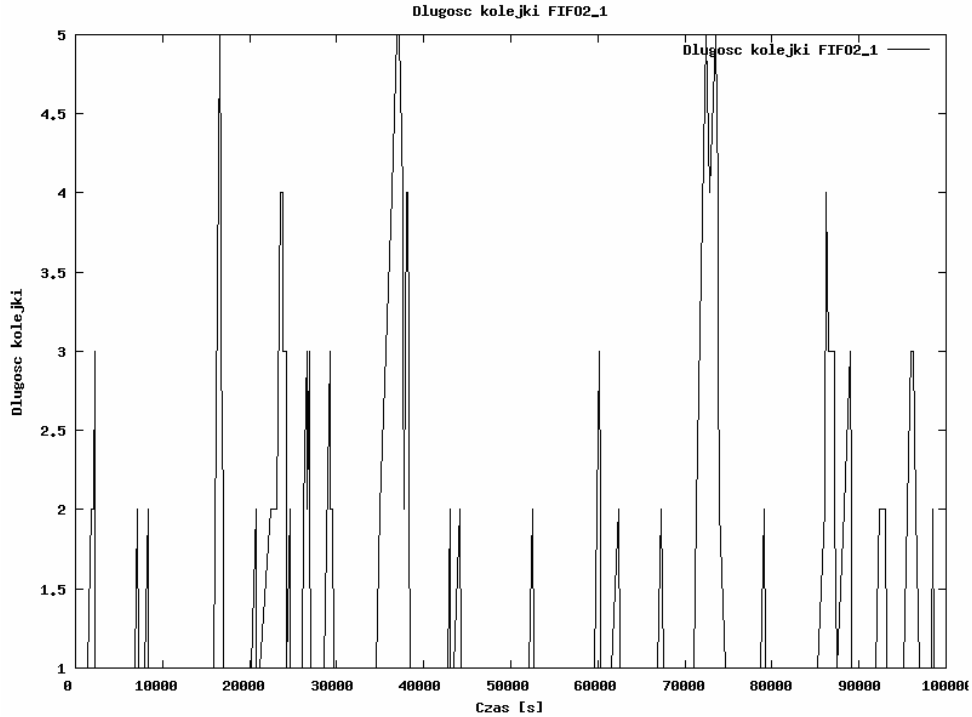

c) Charakterystyka długości kolejki PS1_3 warstwy front-end dla modelu IV (*A*=4, *B*=4) - 500[zapytań/s]



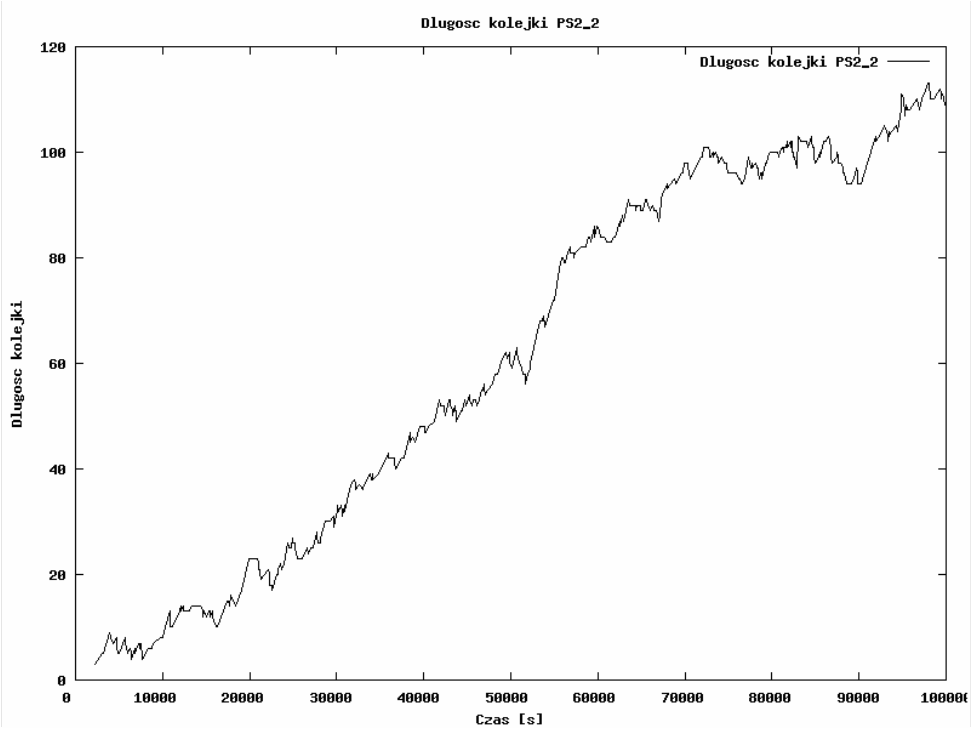

d) Charakterystyka długości kolejki PS1_4 warstwy front-end dla modelu IV (*A*=4, *B*=4) - 500[zapytań/s]



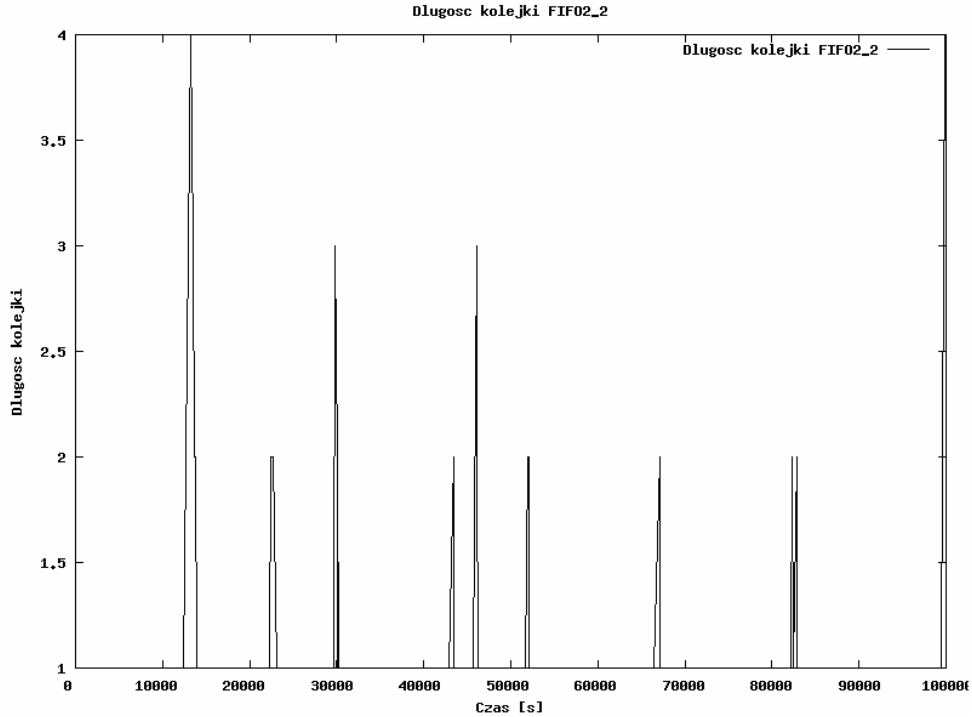

e) Charakterystyka długości kolejki PS2_1 warstwy front-end dla modelu IV (*A*=4, *B*=4) - 500[zapytań/s]



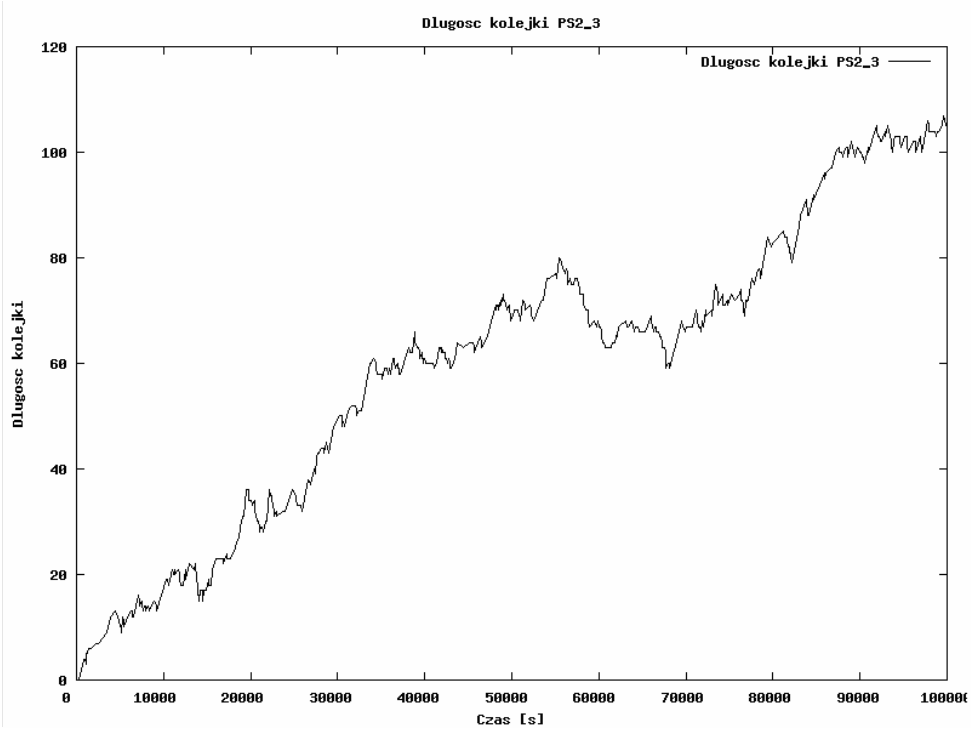

f) Charakterystyka długości kolejki FIFO2_1 warstwy front-end dla modelu IV (*A*=4, *B*=4) - 500[zapytań/s]



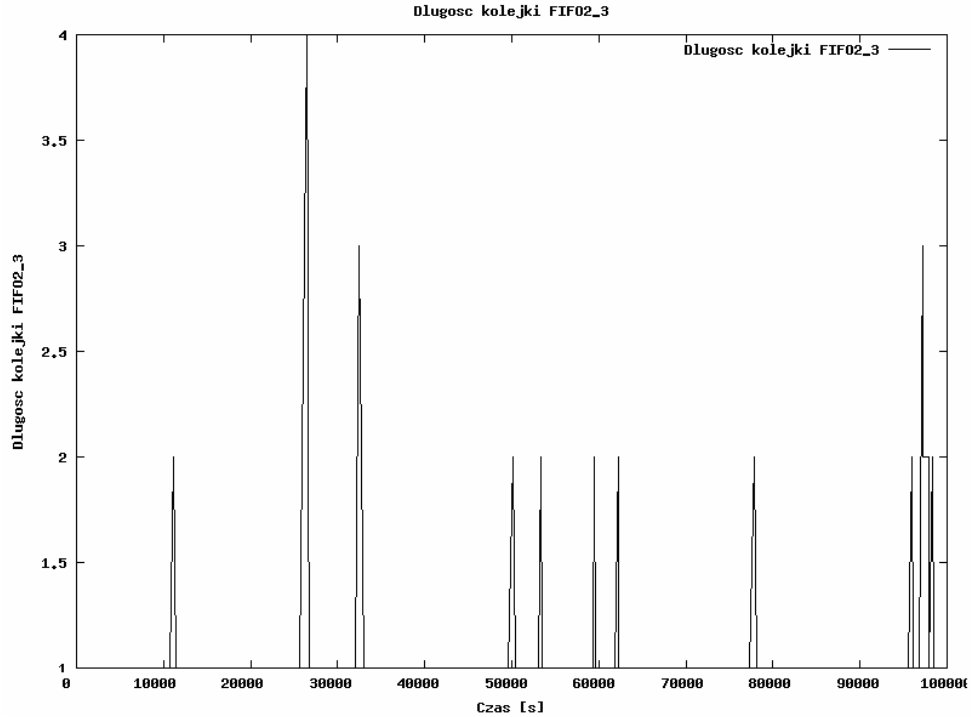

g) Charakterystyka długości kolejki PS2_2 warstwy front-end dla modelu IV (*A*=4, *B*=4) - 500[zapytań/s]



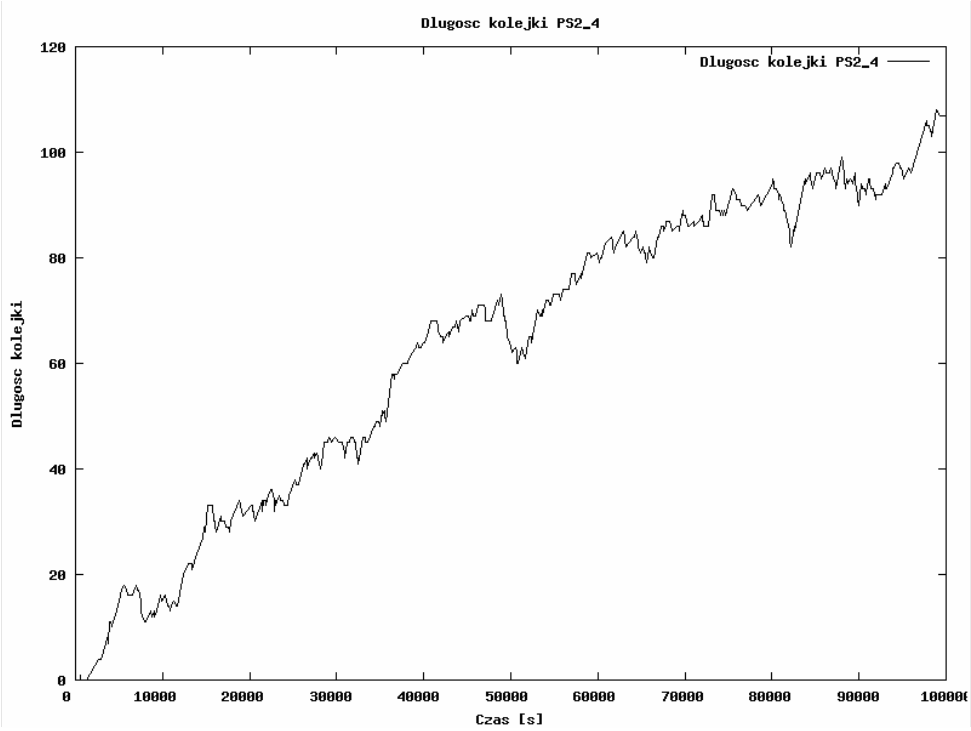

h) Charakterystyka długości kolejki FIFO2_2 warstwy front-end dla modelu IV (*A*=4, *B*=4) - 500[zapytań/s]



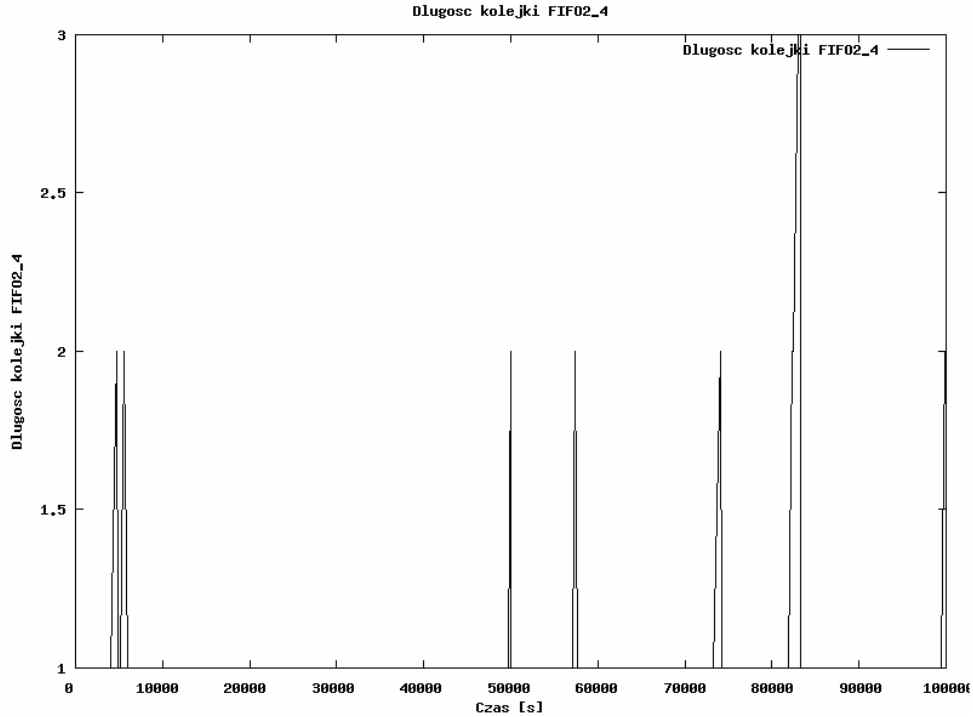

i) Charakterystyka długości kolejki PS2_3 warstwy front-end dla modelu IV (*A*=4, *B*=4) - 500[zapytań/s]




j) Charakterystyka długości kolejki FIFO2_3 warstwy front-end dla modelu IV (*A*=4, *B*=4) - 500[zapytań/s]



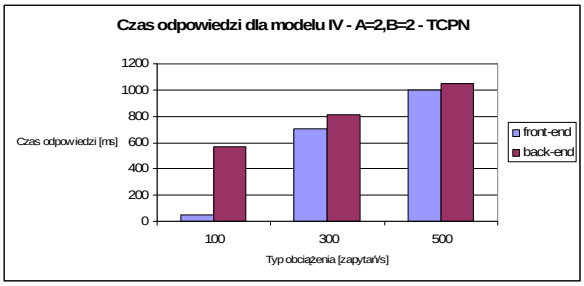

k) Charakterystyka długości kolejki PS2_4 warstwy front-end dla modelu IV (*A*=4, *B*=4) - 500[zapytań/s]



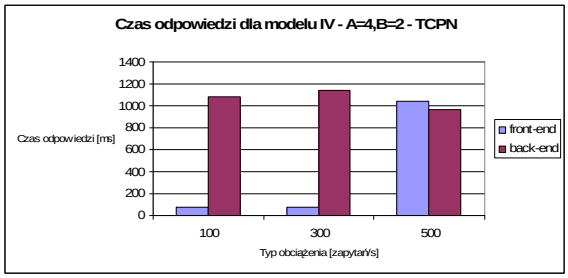

l) Charakterystyka długości kolejki FIFO2_4 warstwy front-end dla modelu IV (*A*=4, *B*=4) - 500[zapytań/s]

Rys. 5.8 Charakterystyki długości kolejek dla modelu IV klastra 4 węzłowego front-end i replikacji 4 węzłowej back-end (500[zapytań/s])

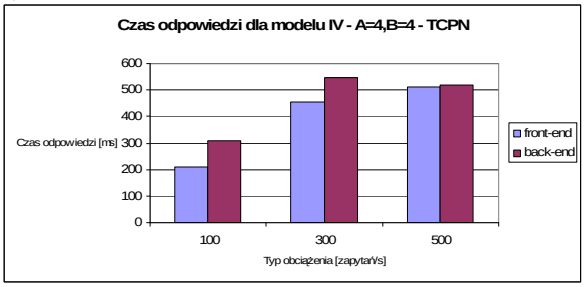
Na wykresach (Rys. 5.9) zaprezentowano czasy odpowiedzi poszczególnych warstw systemu dla modelu IV przy trzech różnych typach obciążenia $\lambda = \{100[1/s], 300[1/s], 500[1/s]\}$. Wykres (Rys. 5.9a) prezentuje przypadek *A*=2, *B*=2, wykres (Rys. 5.9b) prezentuje przypadek *A*=4, *B*=2, natomiast wykres (Rys. 5.9c) prezentuje przypadek *A*=4, *B*=4. Zaobserwować można stopniowe skracanie czasu odpowiedzi warstw przy największym stosowanym obciążeniu szczytowym.




a)



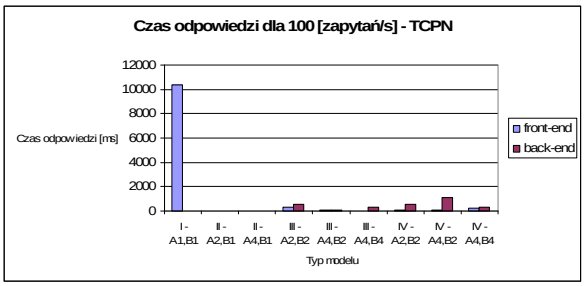

b)



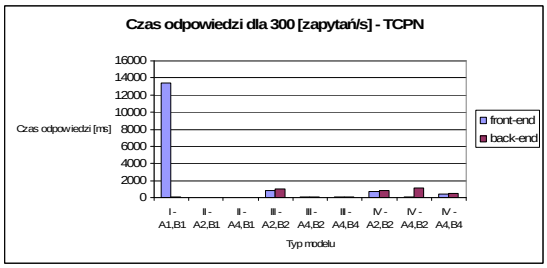

c)

Rys. 5.9 Czasy odpowiedzi warstw (front-end, back-end) przy trzech typach obciążeń dla przykładów modelu IV: a) A=2, B=2, b) A=4, B=2, c) A=4, B=4

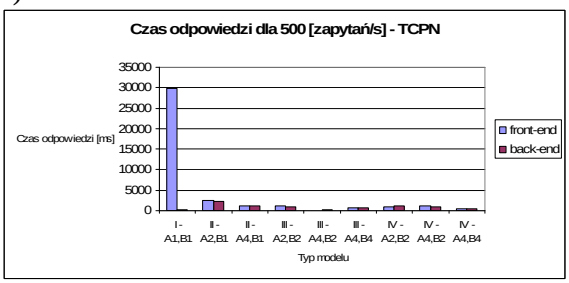
Zaprezentowana szczegółowa analiza modeli TCPN obrazuje tendencje jakie występują w przypadkach zwiększania obciążenia systemu. Zauważyć należy poprawę zrównoważenia systemu w przypadkach zastosowania przetwarzania równoległego w postaci klastra. Użycie rozdzielania zapytań na większą liczbę węzłów powoduje poprawę parametrów wydajnościowych systemu. Zastosowanie replikacji w warstwie *back-end*, nie pociąga za sobą tak wyraźnej poprawy. Jednak jest dobrym posunięciem co można zaobserwować szczególnie w przypadku wzrastającego obciążenia. Rysunki 5.10 prezentują przegląd zachowania wszystkich stosowanych modeli dla poszczególnych obciążeń. Najdłuższe czasy odpowiedzi posiada model I, które wzrastają jeszcze wraz z obciążeniem. Czasy odpowiedzi pozostałych kształtują się podobnie z tendencją malejącą dla modeli z klastrami i replikacją (Rys. 5.10a,b,c).




a)




b)




c)

Rys. 5.10 Czasy odpowiedzi warstw (front-end, back-end) badanych modeli dla trzech typów obciążeń: a) 100[zapytań/s], b) 300[zapytań/s], c) 500[zapytań/s]

## 5.2 ZESTAWIENIE WYNIKÓW TCPN Z CSIM

Chcąc sprawdzić otrzymane wyniki symulacji modeli TCPN utworzono odpowiadające im modele inżynierskie omawianego systemu przy użyciu pakietu CSIM. Porównano ze sobą długości kolejek i czasów odpowiedzi, otrzymanych w trakcie symulacji modelu TCPN z wynikami symulacji CSIM. Symulacje w odpowiadających sobie przypadkach miały identyczne parametry procesów wejściowych i systemów kolejkowych, włączając w to również czas trwania symulacji.

### 5.2.1 PORÓWNANIE DŁUGOŚCI KOLEJEK MODELI TCPN I CSIM

W podrozdziale zamieszczono porównanie zebranych wartości długości kolejek obu symulacji wraz z błędem występującym pomiędzy nimi. Tabele zawierają porównanie wyników średnich długości kolejek modeli symulacyjnych TCPN i CSIM dla badanych modeli ISIROSO. Średni błąd dla wszystkich analizowanych modeli wynosi 15,3%. Średnie błędy odpowiednio dla symulowanych modeli: 12,2% (Tab. 5.1), 8,7% (Tab. 5.2), 15,8% (Tab. 5.3) i 16,7% (Tab. 5.4).[5]

[5] Błąd podawany jest jako orientacyjna wartość w celu zobrazowania różnic w wynikach pomiędzy modelami.

Tab. 5.1 Model I - porównanie wyników symulacji modeli TCPN i CSIM - średnie długości kolejek (średni błąd w analizowanych przykładach wynosi 12,2%)

| $\lambda[1/s]$ | Rodzaj / Kolejka | Model TCPN | Model CSIM | Błąd [%] |
|---|---|---|---|---|
| | Model (I) podstawowy | | | |
| 100 | PS_Q1 | 2149,61 | 1960,89 | 8,8 |
| | PS_Q2 | 0,00 | 0,00 | 0,0 |
| | FIFO_Q2 | 1,00 | 1,00 | 0,0 |
| 300 | PS_Q1 | 14950,90 | 16001,97 | -7,0 |
| | PS_Q2 | 1,52 | 1,63 | -20,3 |
| | FIFO_Q2 | 1,25 | 1,43 | -14,3 |
| 500 | PS_Q1 | 45727,99 | 40034,34 | 12,4 |
| | PS_Q2 | 6,85 | 8,90 | -29,9 |
| | FIFO_Q2 | 5,96 | 7,00 | -17,3 |

Tab. 5.2 Model II - porównanie wyników symulacji modeli TCPN i CSIM - średnie długości kolejek (średni błąd w analizowanych przykładach wynosi 8,7%)

| $\lambda[1/s]$ | Rodzaj / Kolejka | Model TCPN | Model CSIM | Błąd [%] |
|---|---|---|---|---|
| | Model (II) klastra front-end (2 węzły) | | | |
| 100 | PS_Q1_1 | 0,47 | 0,44 | 6,1 |
| | PS_Q1_2 | 0,47 | 0,43 | 7,9 |
| | PS_Q2 | 0,76 | 0,69 | 8,5 |
| | FIFO_Q2 | 1,00 | 1,00 | 0,0 |
| 300 | PS_Q1_1 | 1,37 | 1,37 | 0,2 |
| | PS_Q1_2 | 1,34 | 1,33 | 0,7 |
| | PS_Q2 | 39,79 | 41,32 | -3,8 |
| | FIFO_Q2 | 1,88 | 1,78 | 5,0 |
| 500 | PS_Q1_1 | 20658,96 | 23902,98 | -15,7 |
| | PS_Q1_2 | 20583,79 | 23798,98 | 6,0 |
| | PS_Q2 | 2654,19 | 2839,43 | 7,7 |
| | FIFO_Q2 | 67,37 | 69,00 | 8,5 |
| | Model (II) klastra front-end (4 węzły) | | | |
| 100 | PS_Q1_1 | 0,14 | 0,13 | 9,6 |
| | PS_Q1_2 | 0,16 | 0,14 | 9,0 |
| | PS_Q1_3 | 0,14 | 0,13 | 9,6 |
| | PS_Q1_4 | 0,15 | 0,14 | 7,2 |
| | PS_Q2 | 1,14 | 1,34 | -17,2 |
| | FIFO_Q2 | 1,01 | 1,00 | -0,3 |
| 300 | PS_Q1_1 | 0,32 | 0,22 | 31,6 |
| | PS_Q1_2 | 0,36 | 0,23 | 34,1 |
| | PS_Q1_3 | 0,31 | 0,32 | -5,7 |
| | PS_Q1_4 | 0,31 | 0,30 | 1,5 |
| | PS_Q2 | 36,55 | 39,43 | -7,9 |

| | | | | |
|---|---|---|---|---|
| | FIFO_Q2 | 1,76 | 1,94 | -9,9 |
| 500 | PS_Q1_1 | 413,02 | 411,32 | 0,4 |
| | PS_Q1_2 | 409,37 | 421,18 | -2,9 |
| | PS_Q1_3 | 439,20 | 417,33 | 4,9 |
| | PS_Q1_4 | 500,40 | 413,34 | 17,4 |
| | PS_Q2 | 497,68 | 433,51 | 12,9 |
| | FIFO_Q2 | 12,97 | 15,42 | -18,8 |

Tab. 5.3 Model III - porównanie wyników symulacji modeli TCPN i CSIM - średnie długości kolejek (średni błąd w analizowanych przykładach wynosi 15,8%)

| $\lambda[1/s]$ | Rodzaj / Kolejka | Model TCPN | Model CSIM | Błąd [%] |
|---|---|---|---|---|
| | Model (III) klastra front-end (2 węzły) i back-end (2 węzły) | | | |
| 100 | PS_Q1_1 | 145,95 | 133,32 | 8,6 |
| | PS_Q1_2 | 97,77 | 118,08 | -20,8 |
| | PS_Q2_1 | 526,08 | 498,73 | 5,2 |
| | PS_Q2_2 | 473,23 | 477,92 | -0,9 |
| | FIFO_Q2_1 | 1,00 | 0,92 | 7,4 |
| | FIFO_Q2_2 | 1,00 | 0,76 | 23,8 |
| 300 | PS_Q1_1 | 423,50 | 381,92 | 9,8 |
| | PS_Q1_2 | 361,74 | 372,82 | -3,1 |
| | PS_Q2_1 | 397,38 | 401,83 | -1,1 |
| | PS_Q2_2 | 310,87 | 379,73 | -22,1 |
| | FIFO_Q2_1 | 1,06 | 0,73 | 30,8 |
| | FIFO_Q2_2 | 1,06 | 0,67 | 36,6 |
| 500 | PS_Q1_1 | 1163,06 | 1083,83 | 6,8 |
| | PS_Q1_2 | 1196,08 | 989,82 | 17,2 |
| | PS_Q2_1 | 96,87 | 102,72 | -6,0 |
| | PS_Q2_2 | 94,03 | 99,65 | -5,9 |
| | FIFO_Q2_1 | 1,16 | 0,82 | 29,2 |
| | FIFO_Q2_2 | 1,16 | 0,92 | 20,3 |
| | Model (III) klastra front-end (4 węzły) i back-end (2 węzły) | | | |
| 100 | PS_Q1_1 | 0,28 | 0,18 | 35,9 |
| | PS_Q1_2 | 0,36 | 0,28 | 21,4 |
| | PS_Q1_3 | 0,28 | 0,19 | 30,8 |
| | PS_Q1_4 | 0,25 | 0,18 | 26,6 |
| | PS_Q2_1 | 540,15 | 720,92 | -33,5 |
| | PS_Q2_2 | 631,87 | 654,82 | -3,6 |
| | FIFO_Q2_1 | 1,00 | 0,922 | 7,8 |
| | FIFO_Q2_2 | 1,00 | 0,773 | 22,6 |
| 300 | PS_Q1_1 | 0,80 | 0,68 | 15,0 |
| | PS_Q1_2 | 0,70 | 0,58 | 16,7 |
| | PS_Q1_3 | 0,61 | 0,57 | 6,1 |
| | PS_Q1_4 | 0,75 | 0,66 | 11,7 |
| | PS_Q2_1 | 685,56 | 792,38 | -15,6 |

| | | | | |
|---|---|---|---|---|
| | PS_Q2_2 | 642,73 | 794,32 | -23,6 |
| | FIFO_Q2_1 | 1,05 | 0,91 | 13,7 |
| | FIFO_Q2_2 | 1,04 | 1,03 | 0,8 |
| 500 | PS_Q1_1 | 305,69 | 203,89 | 33,3 |
| | PS_Q1_2 | 450,53 | 290,98 | 35,4 |
| | PS_Q1_3 | 377,68 | 239,38 | 36,6 |
| | PS_Q1_4 | 387,32 | 309,76 | 20,0 |
| | PS_Q2_1 | 131,10 | 109,39 | 16,6 |
| | PS_Q2_2 | 213,27 | 191,39 | 10,3 |
| | FIFO_Q2_1 | 1,39 | 1,49 | -7,7 |
| | FIFO_Q2_2 | 1,38 | 1,27 | 7,6 |
| | Model (III) klastra front-end (4 węzły) i back-end (4 węzły) | | | |
| 100 | PS_Q1_1 | 1,30 | 0,98 | 24,3 |
| | PS_Q1_2 | 1,02 | 0,95 | 6,9 |
| | PS_Q1_3 | 1,44 | 0,99 | 31,3 |
| | PS_Q1_4 | 1,57 | 1,13 | 28,2 |
| | PS_Q2_1 | 57,41 | 40,09 | 30,2 |
| | PS_Q2_2 | 51,97 | 62,83 | -20,9 |
| | PS_Q2_3 | 72,37 | 54,83 | 24,2 |
| | PS_Q2_4 | 49,84 | 55,93 | -12,2 |
| | FIFO_Q2_1 | 1,00 | 0,72 | 27,7 |
| | FIFO_Q2_2 | 1,00 | 0,93 | 6,6 |
| | FIFO_Q2_3 | 1,00 | 0,99 | 0,3 |
| | FIFO_Q2_4 | 1,00 | 0,89 | 10,7 |
| 300 | PS_Q1_1 | 4,31 | 3,90 | 9,5 |
| | PS_Q1_2 | 18,64 | 21,83 | -17,1 |
| | PS_Q1_3 | 13,96 | 12,93 | 7,4 |
| | PS_Q1_4 | 4,67 | 4,22 | 9,5 |
| | PS_Q2_1 | 202,02 | 150,38 | 25,6 |
| | PS_Q2_2 | 190,98 | 177,87 | 6,9 |
| | PS_Q2_3 | 184,59 | 171,82 | 6,9 |
| | PS_Q2_4 | 178,89 | 169,63 | 5,2 |
| | FIFO_Q2_1 | 1,06 | 0,83 | 21,2 |
| | FIFO_Q2_2 | 1,07 | 0,88 | 17,7 |
| | FIFO_Q2_3 | 1,05 | 0,97 | 7,2 |
| | FIFO_Q2_4 | 1,03 | 0,97 | 6,2 |
| 500 | PS_Q1_1 | 226,80 | 301,37 | -14,2 |
| | PS_Q1_2 | 248,97 | 289,83 | -7,4 |
| | PS_Q1_3 | 278,51 | 243,37 | 6,1 |
| | PS_Q1_4 | 243,13 | 242,87 | 0,1 |
| | PS_Q2_1 | 112,14 | 149,39 | -24,5 |
| | PS_Q2_2 | 123,48 | 134,93 | -7,0 |
| | PS_Q2_3 | 119,04 | 114,38 | 2,9 |
| | PS_Q2_4 | 124,71 | 126,39 | -1,0 |
| | FIFO_Q2_1 | 1,49 | 0,99 | 33 |
| | FIFO_Q2_2 | 1,13 | 0,99 | 12,7 |

| | | | | |
|---|---|---|---|---|
| | FIFO_Q2_3 | 1,16 | 0,98 | 15 |
| | FIFO_Q2_4 | 1,96 | 1,29 | 33,8 |

Tab. 5.4 Model IV - porównanie wyników symulacji modeli TCPN i CSIM - średnie długości kolejek (średni błąd w analizowanych przykładach wynosi 16,7%)

| $\lambda[1/s]$ | Rodzaj / Kolejka | Model TCPN | Model CSIM | Błąd [%] |
|---|---|---|---|---|
| | Model (IV) klastra front-end (2 węzły) z replikacją back-end (2 węzły) | | | |
| 100 | PS_Q1_1 | 0,33 | 0,32 | 2,1 |
| | PS_Q1_2 | 0,32 | 0,38 | -18,1 |
| | PS_Q2_1 | 539,61 | 607,84 | -12,6 |
| | PS_Q2_2 | 470,03 | 611,89 | -30,2 |
| | FIFO_Q2_1 | 1,00 | 1,01 | -0,1 |
| | FIFO_Q2_2 | 1,00 | 1,01 | -0,1 |
| 300 | PS_Q1_1 | 471,63 | 398,76 | 15,4 |
| | PS_Q1_2 | 498,25 | 428,43 | 14,1 |
| | PS_Q2_1 | 121,20 | 123,43 | -1,8 |
| | PS_Q2_2 | 240,97 | 159,56 | 33,8 |
| | FIFO_Q2_1 | 1,02 | 1,13 | -10,8 |
| | FIFO_Q2_2 | 1,15 | 1,53 | -32,5 |
| 500 | PS_Q1_1 | 1216,50 | 892,39 | 26,6 |
| | PS_Q1_2 | 1091,10 | 887,43 | 18,7 |
| | PS_Q2_1 | 52,92 | 46,84 | 11,5 |
| | PS_Q2_2 | 77,43 | 54,77 | 29,3 |
| | FIFO_Q2_1 | 1,73 | 1,63 | 5,5 |
| | FIFO_Q2_2 | 1,60 | 1,73 | -7,9 |
| | Model (IV) klastra front-end (4 węzły) z replikacją back-end (2 węzły) | | | |
| 100 | PS_Q1_1 | 0,33 | 0,43 | -30,2 |
| | PS_Q1_2 | 0,32 | 0,32 | 0,6 |
| | PS_Q1_3 | 0,34 | 0,32 | 4,8 |
| | PS_Q1_4 | 0,33 | 0,38 | -13,3 |
| | PS_Q2_1 | 769,22 | 732,32 | 4,8 |
| | PS_Q2_2 | 753,82 | 766,37 | -1,7 |
| | FIFO_Q2_1 | 1,00 | 0,98 | 1,3 |
| | FIFO_Q2_2 | 1,00 | 0,88 | 11,6 |
| 300 | PS_Q1_1 | 0,86 | 0,79 | 7,5 |
| | PS_Q1_2 | 0,79 | 0,88 | -10,8 |
| | PS_Q1_3 | 0,69 | 0,88 | -27,0 |
| | PS_Q1_4 | 1,09 | 0,79 | 27,7 |
| | PS_Q2_1 | 599,35 | 438,94 | 26,8 |
| | PS_Q2_2 | 672,30 | 465,94 | 30,7 |
| | FIFO_Q2_1 | 1,02 | 1,08 | -5,3 |
| | FIFO_Q2_2 | 1,03 | 1,14 | -10,9 |
| 500 | PS_Q1_1 | 441,85 | 543,93 | -23,1 |
| | PS_Q1_2 | 385,13 | 464,44 | -20,1 |

| | | | | |
|---|---|---|---|---|
| | PS_Q1_3 | 466,60 | 548,53 | -17,6 |
| | PS_Q1_4 | 468,85 | 603,49 | -28,7 |
| | PS_Q2_1 | 342,15 | 394,39 | -15,3 |
| | PS_Q2_2 | 271,19 | 344,64 | -27,1 |
| | FIFO_Q2_1 | 1,05 | 1,29 | -22,0 |
| | FIFO_Q2_2 | 1,25 | 1,165 | 7,3 |
| | Model (IV) klastra front-end (4 węzły) z replikacją back-end (4 węzły) | | | |
| 100 | PS_Q1_1 | 514,83 | 476,99 | 7,4 |
| | PS_Q1_2 | 503,24 | 398,89 | 20,7 |
| | PS_Q1_3 | 534,92 | 423,37 | 20,9 |
| | PS_Q1_4 | 515,99 | 441,98 | 14,3 |
| | PS_Q2_1 | 45,11 | 32,73 | 27,4 |
| | PS_Q2_2 | 26,41 | 19,28 | 27,0 |
| | PS_Q2_3 | 36,51 | 37,98 | -4,0 |
| | PS_Q2_4 | 46,00 | 47,36 | -2,9 |
| | FIFO_Q2_1 | 1,17 | 1,48 | -26,8 |
| | FIFO_Q2_2 | 1,95 | 1,79 | 8,5 |
| | FIFO_Q2_3 | 1,35 | 1,79 | -32,5 |
| | FIFO_Q2_4 | 1,82 | 1,69 | 6,8 |
| 300 | PS_Q1_1 | 6,14 | 4,83 | 21,3 |
| | PS_Q1_2 | 4,56 | 3,38 | 25,7 |
| | PS_Q1_3 | 6,16 | 7,93 | -28,7 |
| | PS_Q1_4 | 6,24 | 6,28 | -0,6 |
| | PS_Q2_1 | 100,34 | 67,39 | 32,8 |
| | PS_Q2_2 | 106,48 | 87,29 | 18,0 |
| | PS_Q2_3 | 118,34 | 99,83 | 15,6 |
| | PS_Q2_4 | 117,66 | 82,93 | 29,5 |
| | FIFO_Q2_1 | 1,14 | 1,38 | -36,7 |
| | FIFO_Q2_2 | 1,35 | 1,33 | -28,7 |
| | FIFO_Q2_3 | 1,01 | 1,43 | -29,9 |
| | FIFO_Q2_4 | 1,02 | 1,38 | -7,5 |
| 500 | PS_Q1_1 | 1,61 | 1,43 | 14,9 |
| | PS_Q1_2 | 1,35 | 1,53 | -13,1 |
| | PS_Q1_3 | 1,38 | 1,02 | 25,9 |
| | PS_Q1_4 | 1,67 | 1,63 | 2,4 |
| | PS_Q2_1 | 27,18 | 22,93 | 15,6 |
| | PS_Q2_2 | 40,65 | 29,83 | 26,6 |
| | PS_Q2_3 | 40,83 | 39,83 | 2,5 |
| | PS_Q2_4 | 25,30 | 18,09 | 28,5 |
| | FIFO_Q2_1 | 1,00 | 0,92 | 7,2 |
| | FIFO_Q2_2 | 1,00 | 1,22 | -22,3 |
| | FIFO_Q2_3 | 1,00 | 1,13 | -13,3 |
| | FIFO_Q2_4 | 1,00 | 1,12 | -12,4 |

Zestawienie ze sobą wyników symulacji TCPN i CSIM ma na celu zobrazowanie różnic występujących pomiędzy tymi dwoma sposobami modelowania. Można zauważyć, że

wykonane symulacje nie dały identycznych rezultatów. Wynikać to może z wielu czynników w tym użycia dwóch odmiennych narzędzi symulacji. Głównym powodem są różnice wynikające z implementacji systemów kolejkowych. W przypadku CSIM zostały użyte przygotowane w pakiecie modele systemów kolejkowych i nie ma możliwości ich zmiany. Dla TCPN zostały one utworzone przez autora (Podrozdz. 3.2.4).

### 5.2.2 PORÓWNANIE CZASÓW ODPOWIEDZI MODELI TCPN I CSIM

Kolejnym analizowanym parametrem wydajnościowym jest czas odpowiedzi w poszczególnych warstwach modelu systemu. Czas odpowiedzi prezentuje podobnie jak długości kolejek zachowanie systemu. Zauważyć należy, że otrzymane dla tych samych czasów symulacji - odpowiadające sobie - charakterystyki długości kolejek i czasów odpowiedzi symulacji TCPN i CSIM mają identyczny charakter, dlatego też możliwe jest ich porównywanie.

MODEL PODSTAWOWY (MODEL I)

Analiza wyników symulacji Design/CPN polega na porównaniu w tabeli 5.5 średnich czasów odpowiedzi z wynikami modelu CSIM. Czasy odpowiedzi wahają się od ułamków do nawet kilkudziesięciu sekund. Średni błąd wynosi około 8%.

Tab. 5.5 Model I - porównanie wyników symulacji modeli TCPN i CSIM - średni czas odpowiedzi [ms] (średni błąd w analizowanych przykładach wynosi 8,5%)

| $\lambda[1/s]$ | Rodzaj / Warstwa | Model TCPN | Model CSIM | Błąd [%] |
|---|---|---|---|---|
| | Model (I) podstawowy | | | |
| 100 | front-end | 10400 | 10240 | 1,5 |
| | back-end | 27 | 34 | -25,9 |
| 300 | front-end | 13445 | 14852 | -10,5 |
| | back-end | 135 | 139 | -2,9 |
| 500 | front-end | 29850 | 29892 | -0,1 |
| | back-end | 275 | 302 | -9,8 |

MODEL KLASTRA FRONT-END (MODEL II)

Uwidocznione w tabeli 5.6 wyniki symulacji potwierdzają wcześniejsze rozważania. Zastosowanie klastrowania w warstwie *front-end* jest dobrym posunięciem. Problem wydajności pojawia się przy większej liczbie zapytań na sekundę. Średni błąd między modelami (Tab. 5.6) wynosi około 5%.

Tab. 5.6 Model II - porównanie wyników symulacji modeli TCPN i CSIM - średni czas odpowiedzi [ms] (średni błąd w analizowanych przykładach wynosi 5,5%)

| $\lambda[1/s]$ | Rodzaj / Kolejka | Model TCPN | Model CSIM | Błąd [%] |
|---|---|---|---|---|
| | Model (II) klastra front-end (2 węzły) | | | |
| 100 | front-end | 1 | 1 | 0,0 |
| | back-end | 1 | 1 | 0,0 |
| 300 | front-end | 19 | 24 | -26,3 |
| | back-end | 21 | 25 | -19,0 |
| 500 | front-end | 24825 | 24847 | -0,1 |
| | back-end | 21480 | 21322 | 0,7 |
| | Model klastra front-end (4 węzły) | | | |
| 100 | front-end | 1 | 1 | 0,0 |
| | back-end | 1 | 1 | 0,0 |
| 300 | front-end | 25 | 26 | -4,0 |
| | back-end | 44 | 39 | 11,4 |
| 500 | front-end | 1017 | 1003 | 1,4 |
| | back-end | 1073 | 1102 | -2,7 |

MODEL KLASTRÓW OBU WARSTW (MODEL III)

Podobnie jak w poprzednich modelach symulacje CSIM odpowiadają wynikom TCPN (Tab. 5.7). Czasy odpowiedzi całego systemu nawet w przypadku 500[zapytań/s] oscylują w granicach sekundy. Średni błąd między modelami wynosi około 10%.

Tab. 5.7 Model III - porównanie wyników symulacji modeli TCPN i CSIM - średni czas odpowiedzi [ms] (średni błąd w analizowanych przykładach wynosi 9,6%)

| $\lambda[1/s]$ | Rodzaj / Kolejka | Model TCPN | Model CSIM | Błąd [%] |
|---|---|---|---|---|
| | Model (III) klastra front-end (2 węzły) i back-end (2 węzły) | | | |
| 100 | front-end | 305 | 391 | 28,2 |
| | back-end | 521 | 507 | -2,7 |
| 300 | front-end | 832 | 1056 | -26,9 |
| | back-end | 1014 | 1121 | -10,5 |
| 500 | front-end | 1131 | 1176 | -3,9 |
| | back-end | 812 | 513 | -29,9 |
| | Model (III) klastra front-end (4 węzły) i back-end (2 węzły) | | | |
| 100 | front-end | 34 | 34 | 0,0 |
| | back-end | 312 | 352 | -12,8 |
| 300 | front-end | 58 | 54 | 6,9 |
| | back-end | 67 | 58 | 13,4 |
| 500 | front-end | 649 | 632 | 2,6 |
| | back-end | 728 | 672 | 7,7 |

| | Model (III) klastra front-end (4 węzły) i back-end (4 węzły) | | | |
|---|---|---|---|---|
| 100 | front-end | 51 | 65 | -27,4 |
| | back-end | 88 | 91 | -0,4 |
| 300 | front-end | 59 | 63 | -6,8 |
| | back-end | 73 | 62 | 2,9 |
| 500 | front-end | 110 | 120 | -8,6 |
| | back-end | 131 | 117 | 10,7 |

MODEL KLASTRA I REPLIKACJI (MODEL IV)

Wyniki obu symulacji potwierdzają ponownie słuszność stosowanego podejścia (Tab. 5.8). Zastosowanie replikacji nie powoduje drastycznego obniżenia czasu odpowiedzi jednak zahamowany zostaje proces jego wydłużania. Średni błąd pomiędzy wynikami modeli wynosi niecałe 10%.

Tab. 5.8 Model IV - porównanie wyników symulacji modeli TCPN i CSIM - średni czas odpowiedzi [ms] (średni błąd w analizowanych przykładach wynosi 9,5%)

| $\lambda[1/s]$ | Rodzaj / Kolejka | Model TCPN | Model CSIM | Błąd [%] |
|---|---|---|---|---|
| | Model (IV) klastra front-end (2 węzły) z replikacją back-end (2 węzły) | | | |
| 100 | front-end | 49 | 50 | -2,0 |
| | back-end | 567 | 598 | -5,5 |
| 300 | front-end | 701 | 743 | -5,9 |
| | back-end | 813 | 798 | 1,8 |
| 500 | front-end | 10011 | 11091 | -10,8 |
| | back-end | 10508 | 10831 | -3,1 |
| | Model (IV) klastra front-end (4 węzły) z replikacją back-end (2 węzły) | | | |
| 100 | front-end | 75 | 73 | 2,7 |
| | back-end | 1083 | 867 | 19,9 |
| 300 | front-end | 79 | 85 | -7,6 |
| | back-end | 1144 | 989 | 13,5 |
| 500 | front-end | 1042 | 1191 | -14,3 |
| | back-end | 965 | 950 | -9,8 |
| | Model (IV) klastra front-end (4 węzły) z replikacją back-end (4 węzły) | | | |
| 100 | front-end | 210 | 282 | -34,2 |
| | back-end | 308 | 290 | 5,8 |
| 300 | front-end | 454 | 472 | -3,9 |
| | back-end | 545 | 499 | 8,4 |
| 500 | front-end | 512 | 595 | -16,2 |
| | back-end | 520 | 384 | 5,2 |

## 5.3 WNIOSKI Z ANALIZY

Szczegółowe wnioski z analizy modeli ISIROSO określają reakcje poszczególnych konfiguracji na obciążenie. Wzrost liczby elementów warstwy *front-end* (zastosowanie klastra) powoduje przyspieszenie realizowanych obliczeń. Dla mniejszych obciążeń rozwiązanie takie daje pozytywne efekty. Jednak wzrost liczby zapytań powoduje dalsze wydłużenie czasu odpowiedzi. Zastosowanie klastra w warstwie *front-end* i *back-end* skutkuje, podobnie jak w poprzednim przypadku, skróceniem czasu odpowiedzi warstwy *front-end* przy jednoczesnym wydłużeniu czasu odpowiedzi w warstwie *back-end*. Poprawia nieco ten niekorzystny stan powiększenie klastra warstwy *back-end*, ale nie są to zmiany znaczące. Dopiero zastosowanie podobnej liczby węzłów replikacji warstwy *back-end* w porównaniu z liczbą węzłów klastra *front-end* daje bardziej zadawalający efekt we wszystkich przypadkach. Zastosowanie replikacji pociąga za sobą konieczność synchronizacji, jednak jej koszt jest mniejszy od zysku jaki otrzymuje się dzięki dodatkowym stanowiskom obsługi. Obciążenie poszczególnych węzłów nieco wzrasta, ale globalnie system skraca czas odpowiedzi dzięki równomiernemu rozłożeniu zapytań. W przypadkach, gdzie obserwowane jest niezrównoważenie, występują np. bardzo krótkie kolejki jednej warstwy i jednocześnie bardzo długie kolejki innej. Brak zrównoważenia wiąże się z brakiem równomiernego rozłożenia obciążeń w warstwach i pomiędzy nimi.

Generalnie zrównoważenie systemów w przypadku zwiększającego się obciążenia, ulega zachwianiu. Kolejne zwiększanie zewnętrznego strumienia zgłoszeń jedynie pogarsza wydajność. Możliwa jest oczywiście modyfikacja, polegająca na rozbudowie elementów sprzętowych serwerów w celu przyspieszenia ich przetwarzania. Rozwiązanie to wiąże się z kosztami i koniecznością wymiany całych serwerów np. ze względu na niekompatybilność sprzętu. Dążyć więc należy do systemów zdecentralizowanych, które są bardziej naturalne dla systemów internetowych. Opracowane modele systemów zbudowane z klasterów i replikacji pozwalają na skrócenie czasu odpowiedzi i doprowadzenie w konsekwencji do jego zrównoważenia. Przy założeniu, że do budowy systemów używa się identycznych komputerów, skierowano wysiłki na określenie sposobu ich połączenia w kierunku osiągnięcia założonego celu. Na podstawie przeprowadzonej analizy można wnioskować, że celowe jest zastosowanie modyfikacji budowy systemów bez ingerencji w budowę komputera i oprogramowanie.

W celu osiągnięcia zrównoważenia systemów konieczne jest więc:

- zastosowanie warstw w budowie systemów, w celu rozdzielenia przetwarzanych zapytań,
- dobór, zależnie od występującego obciążenia, liczby węzłów w warstwie *front-end* struktury klastra,
- zastosowanie replikacji z liczbą węzłów sięgającą powyżej połowy liczby węzłów *front-end*,
- użycie do budowy systemów identycznych komputerów umożliwiających konstrukcję prostszych algorytmów rozdzielania zapytań według najlepiej tych samych zasad dla wszystkich węzłów.

# 6 EKSPERYMENTALNA WERYFIKACJA MODELI

Celem rozdziału jest weryfikacja poprawności przyjętych modeli ISIROSO. Eksperymenty mają na celu sprawdzenie proponowanych rozwiązań w przypadku rzeczywistej implementacji interaktywnych systemów internetowych realizujących obsługę szybkozmiennych ofert.

Na przygotowanym środowisku eksperymentalnym użyto wybranych obciążeń referencyjnego systemu giełdy internetowej. W wyniku eksperymentów otrzymano czasy odpowiedzi poszczególnych warstw dla trzech przykładowych typów obciążeń. W środowisku laboratoryjnym dokonano sprawdzenia używanego rozkładu wykładniczego do modelowania procesu przybywania zapytań. Identyczne charakterystyki strumienia zgłoszeń użyto do symulacji modeli z wykorzystaniem Design/CPN i CSIM. Węzły systemu rzeczywistego odpowiadają węzłom zawartym w proponowanych modelach ISIROSO. Końcowym elementem prac jest porównanie wyników czasów odpowiedzi poszczególnych warstw systemu dla eksperymentów i obu symulacji.

## 6.1 PARAMETRY MODELU BIZNESOWEGO

Giełda internetowa jest złożonym systemem, dlatego istotne jest utworzenie systemu referencyjnego realizującego przedstawione funkcje. Ważnym składnikiem systemu, którego budowę należy dokładnie przeanalizować jest moduł generujący zapytania klientów. Istnieją dwie metody modelowania zachowań klientów [126]. Pierwsza z nich polega na odtwarzaniu zachowań klientów z „logów" prawdziwych serwerów (np. [141, 142]). „Log" jest plikiem rejestru zapytań wysyłanych do systemu rzeczywistego. Drugim sposobem jest budowa generatora zapytań [72], który stosując odpowiednie rozkłady generuje ruch. W [28] zaprezentowano przykład zastosowania odpowiednich rozkładów wraz z ich parametrami.

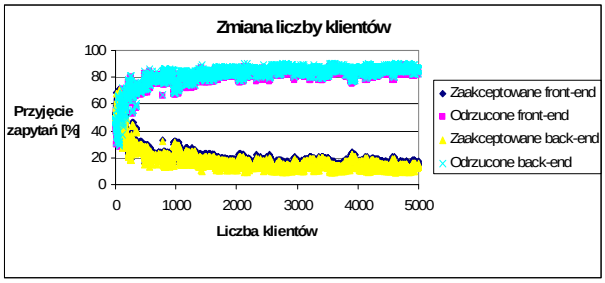
W celu określenia parametrów uproszczonego systemu referencyjnej giełdy internetowej i jej klienta zbudowano model biznesowy. Wyznaczenie tych parametrów pozwala na ustalenie zasad działania referencyjnej giełdy internetowej. Model biznesowy wzorowany jest na formalnym modelu giełdy uproszczonej bazującej na WGPW. W modelu biznesowym skupiono się głównie na parametrach ilościowych wykorzystanych w eksperymentach z referencyjną giełdą internetową.

Na wykresach w tym rozdziale przedstawione zostaną wyniki wybranych symulacji i eksperymentów dla zadanych parametrów ilościowych (Tab. 6.1):

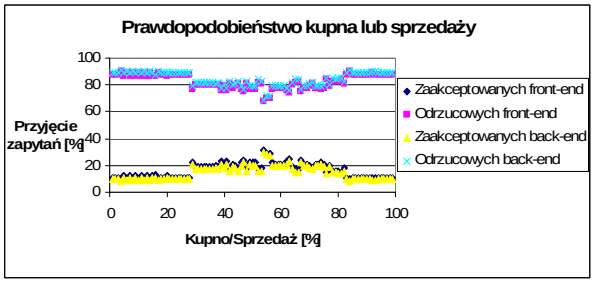
- liczba klientów - oznacza liczbę graczy (Rys. 6.1a), którzy składają zapytania (grają na giełdzie),
- prawdopodobieństwo kupna lub sprzedaży - wartość procentowa stosunku zleceń kupna do zleceń sprzedaży (Rys. 6.1b),
- liczba akcji powodująca zamianę oferty - procentowa liczba akcji wykupywanych lub wysprzedawanych (Rys. 6.1c), po której następuje modyfikacja ceny akcji (szybkozmienność oferty),
- liczba spółek - których akcje wystawione są na giełdzie w postaci oferty (Rys. 6.2),

- akceptowana zmiana oferty - procentowa wartość określająca zmianę ceny akcji w ofercie systemu (Rys. 6.3a,b,c,d), którą klient akceptuje jako dozwoloną i pozostawia swoje zlecenie nadal aktualne,

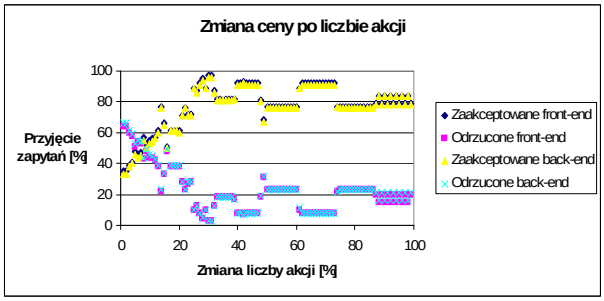
Parametry określają warunki działania giełdy referencyjnej i jej klienta.




a)

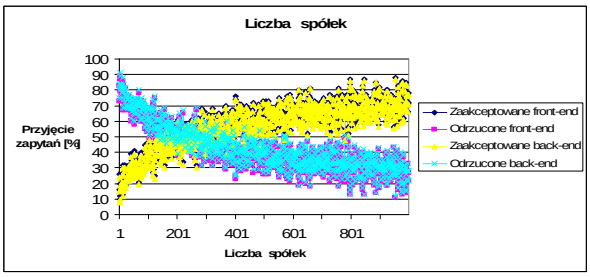



b)

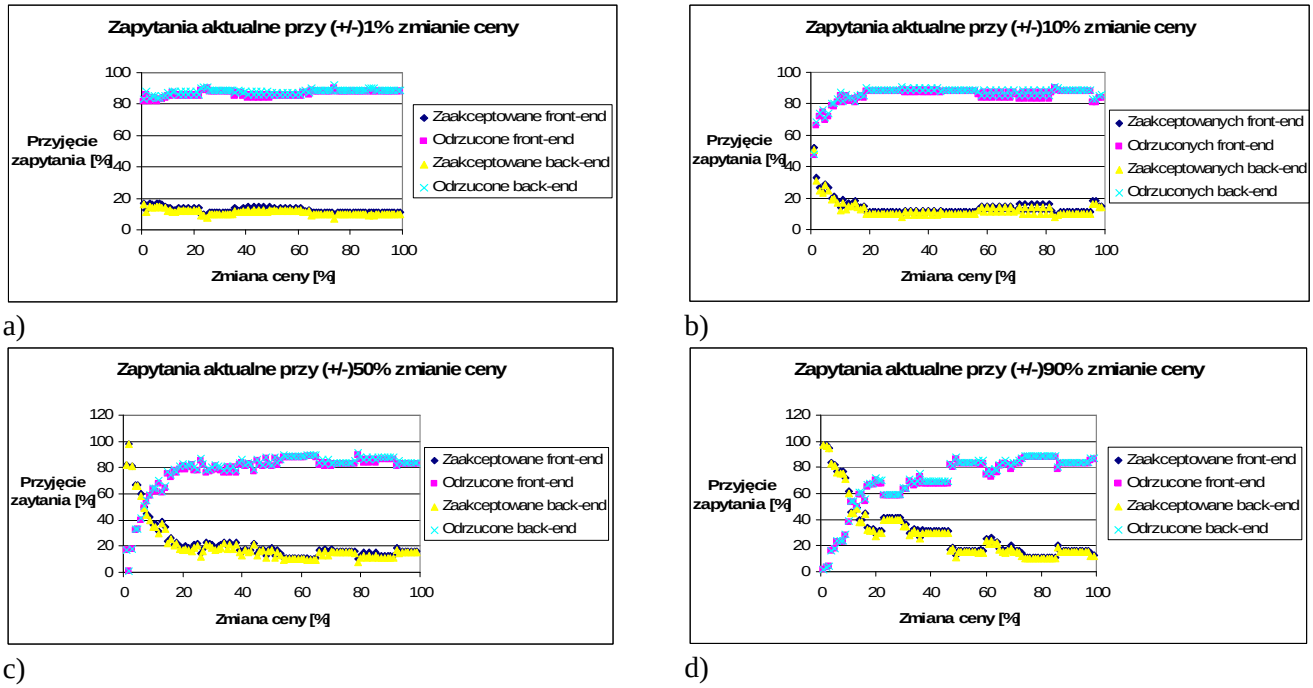



c)

Rys. 6.1 Wykresy: a) zmiany liczby klientów, b) prawdopodobieństwa kupna lub sprzedaży, c) liczby akcji kupna/sprzedaży po której następuje modyfikacja oferty




Rys. 6.2 Wykres liczby spółek




Rys. 6.3 Wykresy akceptowanej zmiany oferty do zrealizowania zlecenia: a) (+/-)1%, b) (+/-)10%, c) (+/-)50%, d) (+/-)90%

Tab. 6.1 Parametry modelu biznesowego giełdy internetowej

| Wykres / Parametr | Rys. 6.1a | Rys. 6.1b | Rys. 6.1c | Rys. 6.2 | Rys. 6.3a | Rys. 6.3b | Rys. 6.3c | Rys. 6.3d | Środowisko eksperymentalne |
|---|---|---|---|---|---|---|---|---|---|
| Liczba klientów | 1-3000 | 1-1000 | 1-1000 | 1-1000 | 1-1000 | 1-1000 | 1-1000 | 1-1000 | 1-1000 |
| Prawdopodobieństwo kupna/sprzedaży [%/%] | 50/50 | od 0/100 do 100/0 | 50/50 | 50/50 | 50/50 | 50/50 | 50/50 | 50/50 | 50/50 |
| Liczba akcji powodująca zmianę oferty [(+/-)%] | 10 | 10 | 1-100 | 10 | 10 | 10 | 10 | 10 | 10 |
| Akceptowana zmiana oferty [(+/-)%] | 1-10 | 1-10 | 1-10 | 1-10 | 1 | 1-10 | 1-50 | 1-90 | 10 |
| Liczba spółek | 1-200 | 1-200 | 1-200 | 1-1000 | 1-200 | 1-200 | 1-200 | 1-200 | 1-200 |

W wyniku przeprowadzonych analiz zdefiniowano parametry dla eksperymentów w środowisku laboratoryjnym referencyjnej giełdy internetowej, którymi są wartości losowe z przedziałów (Tab. 6.1).

## 6.2 ŚRODOWISKO EKSPERYMENTALNE

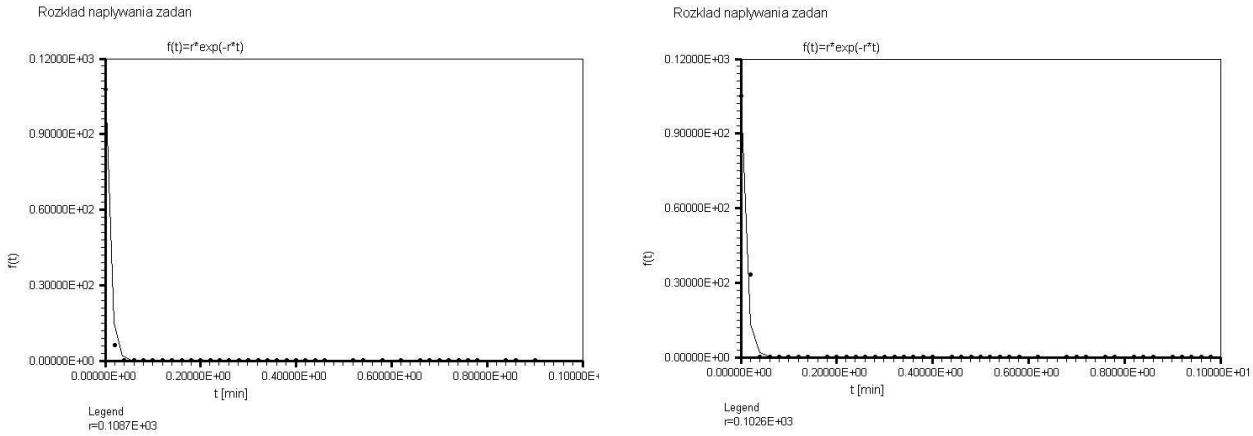
W systemie referencyjnym wprowadzono, zgodnie z założeniami (Podrozdz. 3.2):

- ograniczenie wielkości zlecenia - czyli określenie maksymalnej liczby akcji (51% liczby akcji spółki) jaką możne kupić klient w jednym zapytaniu,
- sprawdzenie liczby dostępnych akcji - sprawdzenie w przypadku kupna, czy liczba akcji którą gracz chce kupić jest dostępna,
- sprawdzenie dostępności środków płatniczych - określenie, w przypadku kupna, czy klient posiada środki płatnicze na zrealizowanie transakcji,
- sprawdzenie liczby posiadanych akcji - czyli sprawdzenie, w przypadku sprzedaży, liczby akcji posiadanych przez gracza,

Inwestor składający zlecenie określa:

- rodzaj zapytania (kupno/sprzedaż),
- spółkę,
- wartość,
- liczbę akcji.

Do analiz wybrano trzy typy obciążenia: 100[zapytań/s], 300[zapytań/s] i 500[zapytań/s]. Wybrano je na podstawie przeprowadzonych własnych analiz i informacji opublikowanych w [48]. Poszerzono je o ostatni typ (500[zapytań/s]) traktowany jako ekstremalne obciążenie systemu giełdowego. Obciążenia generowane było przy wykorzystaniu programu Symulatora Obciążenia Klientów (SOK) [60, 61]. Weryfikacji poprawności użytego obciążenia dokonano, dzięki plikom rejestracji zdarzeń „log" dla systemu podstawowego. Każda linia rejestru zdarzeń oznacza zapytanie i zawiera datę określającą czas zgłoszenia zapytania, rozmiar oraz opis żądanego dokumentu. Trudno uzyskać rzeczywisty rejestr zdarzeń z systemu dlatego zdecydowano się na automatyczne wygenerowanie obciążenia, przy wykorzystaniu aplikacji testującej (SOK). Wybrane sprawdzenia dla 100[zapytań/s] pokazano na rysunkach 6.4a,b. Do tego sprawdzenia wybrano programy: *CurveExpert* [135] (wykorzystany również m.in. w [115]) i *LAB Fit* [140]. $\lambda$ jest parametrem rozkładu wykładniczego równym w prezentowanym przypadku (Rys. 6.4) dla procesu wejścia $\lambda = 108{,}7[1/s]$ (test1) i $\lambda = 102{,}6[1/s]$ (test2). Zgodnie z pracami [48, 115] zastosowane rozkłady napływania zapytań mają więc charakter wykładniczy. Otrzymane rozkłady rozmiarów napływających zapytań również miały charakter wykładniczy. Pominięto je jednak, podobnie jak w innych pracach [50, 51]. Wpływ rozmiarów napływających zapytań na parametry wydajnościowe jest znikomo mały czego przykładem może być [115], gdzie podaje się wpływ (1-2%). W dalszych rozważaniach głównym parametrem opisującym strumienie zgłoszeń będzie rozkład czasu przybywania zapytań. Na podstawie wyników eksperymentów dla pojedynczego serwera ustalono również charakter procesu obsługi użyty w symulacjach. Dostrojenie polegało na wyskalowaniu pojedynczego serwera tak, aby przetwarzał on generowane zapytania tworząc identyczną charakterystykę rozkładów czasów odpowiedzi jak symulator. Dokonano tego dzięki wymuszeniu sztucznego limitu przepustowości interfejsu sieciowego [64, 82. 91].

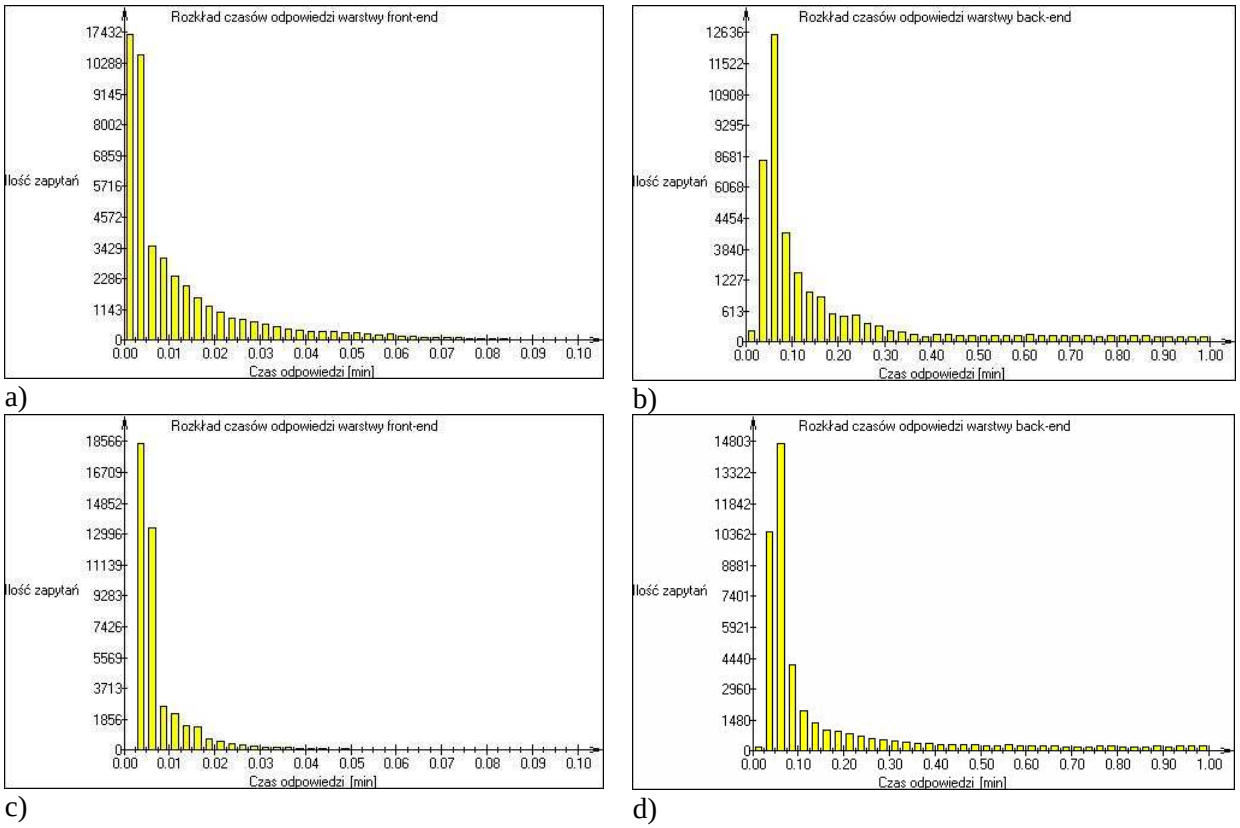



a) b)

Rys. 6.4 Weryfikacja (program *LAB Fit*) rozkładu strumienia zapytań z rejestru zdarzeń: a) test1, b) test2[6]

Rozkłady czasów odpowiedzi symulacji dla poszczególnych warstw (Rys. 6.5a,b) zostały zweryfikowane z odpowiadającymi im charakterystykami (Rys. 6.5c,d) otrzymanymi w wyniku eksperymentów. Przykład z rysunku 6.5 prezentuje model I systemu przy $\lambda = 3[1/s]$ i $\mu = 10[1/s]$. Czas eksperymentu był taki sam jak czas symulacji i wynosił - podobnie jak w rozdziale 5 - 100000[s]. Widać, że najwięcej zapytań było obsłużonych w krótkich przedziałach czasu.

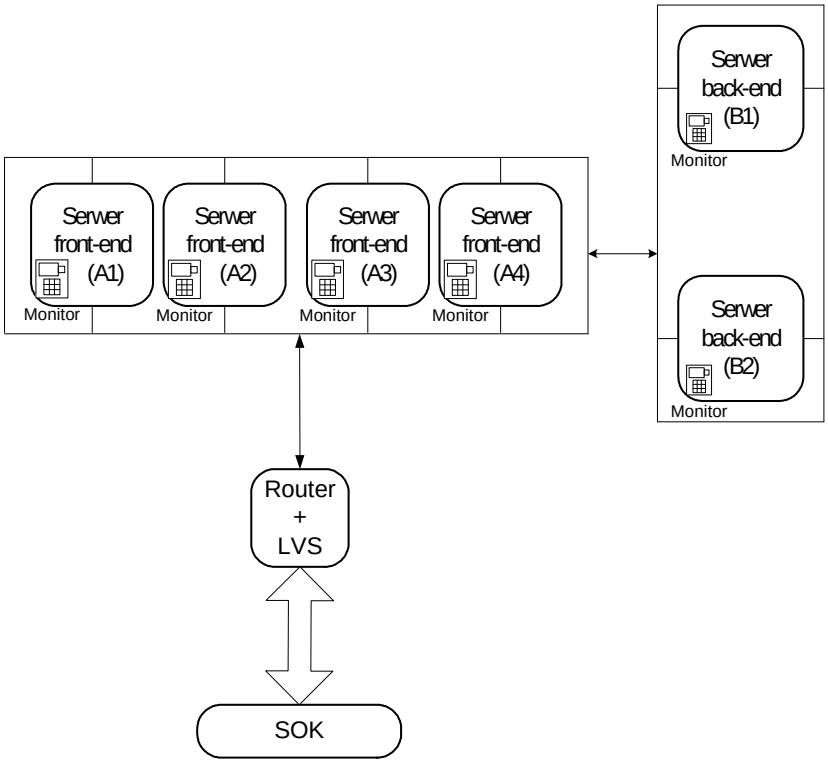
[6] *r* jest parametrem rozkładu wykładniczego wykorzystywanym w programie LAB Fit - $r = \lambda$.




Rys. 6.5 Rozkłady czasów odpowiedzi - model I (przykład): symulator TCPN - a) warstwa front-end, b) warstwa back-end oraz eksperymentów systemu referencyjnego - c) warstwa front-end, d) warstwa back-end

## 6.3 REALIZACJA FIZYCZNA SYSTEMU IGA

W celu pozyskania wyników do analiz symulacyjnych konieczne jest zbudowanie rzeczywistego systemu w warunkach laboratoryjnych. Do budowy Internetowej Giełdy Akcji (IGA) użyto bezpłatnych usług i narzędzi dostępnych za darmo na platformę systemu operacyjnego Linux [62].




Rys. 6.6 Schemat środowiska IGA

Środowisko laboratoryjne IGA składa się z (Rys. 6.6):

- segmentu sieci (1Gb/s) [82, 92, 103],
- 6 węzłów (komputer: płyta i945, procesor Intel Core 2 Duo E6400, pamięć 1GB RAM),
- rozdzielacz obciążenia (komputer: płyta SE7520BD2, procesor Intel XEON, pamięć 2GB RAM, LVS),
- skonfigurowanego systemu operacyjnego 64-bitowy SuSE Linux 10 (jądro 2.6.13) [64, 91],
- SOK (Perl),
- monitora plików rejestru zdarzeń „log".

System IGA jest dostępny (dla wybranych IP) pod adresem http://gielda.prz-rzeszow.pl/online/ (Rys. 6.6), a składa się fizycznie z (Rys. 6.7):

- router+LVS+DNS (zawierający „zaporę sieciową" (*iptables*)): interfejs zewnętrzny - 62.93.34.170 (interfejs *eth0*) i interfejs wewnętrzny - 10.10.2.1 (interfejs *eth1*),
- serwery *front-end* (prezentacji - Apache+PHP): 62.93.34.155 (10.10.2.155), 62.93.34.156 (10.10.2.156), 62.93.34.157 (10.10.2.157), 62.93.34.158 (10.10.2.158),
- serwery *back-end* (bazodanowe - MySQL): 62.93.34.159 (10.10.2.159), 62.93.34.160 (10.10.2.160),
- SOK: 62.93.34.161.

Dla systemu IGA zastosowano rozdzielanie obciążania przy użyciu oprogramowania LVS bazującego na *ipvsadm* i różnych procedurach szeregowania dla obu warstwy. Replikacja

w warstwie *back-end* polegała na użyciu rzeczywistego systemu z jednostronną asynchroniczną replikacją w MySQL dla dwóch węzłów [83].

Rys. 6.7 Sprzętowa realizacja systemu IGA

Szczegóły implementacji systemu uproszczonej giełdy internetowej znaleźć można w dodatku. Na zrzutach ekranu podanych na rysunkach 6.8a,b zaprezentowano wygląd IGA i możliwe interakcje inwestora. Przedstawiony system giełdy internetowej ogranicza się do notowań ciągłych i do jedynie niezbędnych operacji klienta związanych z realizacją zleceń. Występuje tu więc logowanie do systemu, kupowanie lub sprzedawanie akcji czy operacje finansowe. Giełda wizualizuje zmiany cen i liczby akcji, obrót w czasie trwania sesji, itp. Klient uzyskuje dostęp do podstawowych informacji o stanie akcji i swojego portfela, co umożliwia mu podejmowanie decyzji inwestorskich.


a)


b)

Rys. 6.8 Internetowa giełda akcji: a) widok ogólny, b) widok po zalogowaniu

# 6.4 WERYFIKACJA MODELI PRZY UŻYCIU ŚRODOWISKA EKSPERYMENTALNEGO

Weryfikacji podlegają wyniki otrzymane w czasie symulacji modeli TCPN i CSIM z eksperymentami w środowisku IGA. Parametrem użytym do porównania jest średni czas odpowiedzi w poszczególnych warstwach. Weryfikacji i analizy dokonano przy różnych obciążeniach napływających zapytań dla wybranych modeli ISIROSO.

## 6.4.1 CZASY ODPOWIEDZI

MODEL PODSTAWOWY (MODEL I)

Potwierdzeniem wyników symulacji Design/CPN jest tabela 6.2, obrazująca średnie czasy odpowiedzi systemu dla TCPN i eksperymentów w systemie referencyjnym giełdy internetowej. Czasy odpowiedzi modelu TCPN wahają się od 27[ms] do nawet 29[s] w pojedynczej warstwie i odpowiednio dla eksperymentów od 29[ms] do około 27[s]. Średni orientacyjny błąd symulacji TCPN i eksperymentów wynosi około 7%.

Tab. 6.2 Model I - porównanie wyników symulacji TCPN i eksperymentów dla średniego czasu odpowiedzi [ms] (średni błąd w weryfikowanych przykładach wynosi 6,7%)

| Obciążenie [zapytań/s] | Rodzaj / Warstwa | Model TCPN | Eksperymenty | Błąd [%] |
|---|---|---|---|---|
| | Model (I) podstawowy | | | |
| 100 | front-end | 10400 | 10756 | -3,4 |
| | back-end | 27 | 29 | -7,4 |
| 300 | front-end | 13445 | 11282 | 16,1 |
| | back-end | 135 | 129 | 4,4 |
| 500 | front-end | 29850 | 27832 | 6,8 |
| | back-end | 275 | 269 | 2,2 |

MODEL KLASTRA FRONT-END (MODEL II)

Prezentowane w tabeli 6.3 wyniki eksperymentów IGA i wyniki symulacji pozytywnie weryfikują poprawność modelu TCPN klastra *front-end*. Średni błąd pomiędzy symulacją i eksperymentami wynosi 15%.

Tab. 6.3 Model II - porównanie wyników symulacji TCPN i eksperymentów dla średniego czasu odpowiedzi [ms] (średni błąd w weryfikowanych przykładach wynosi 15,0%)

| Obciążenie [zapytań/s] | Rodzaj / Warstwa | Model TCPN | Eksperymenty | Błąd [%] |
|---|---|---|---|---|
| | Model (II) klastra front-end (2 węzły) | | | |
| 100 | front-end | 1 | 1 | 0,0 |
| | back-end | 1 | 1 | 0,0 |
| 300 | front-end | 19 | 22 | -15,7 |
| | back-end | 21 | 15 | 28,5 |
| 500 | front-end | 2482 | 1808 | 27,1 |
| | back-end/500 | 2148 | 1776 | 17,3 |

| | Model (II) klastra front-end (4 węzły) | | | |
|---|---|---|---|---|
| 100 | front-end | 1 | 1 | 0,0 |
| | back-end | 1 | 1 | 0,0 |
| 300 | front-end | 25 | 32 | -28,0 |
| | back-end | 44 | 34 | 22,7 |
| 500 | front-end | 1017 | 898 | 11,7 |
| | back-end | 1073 | 762 | 28,9 |

MODEL KLASTRÓW OBU WARSTW (MODEL III)

Dla modeli z klastrami obu warstw uzyskano wysoką zgodność (Tab. 6.4). Średni błąd jest nieco wyższy i wynosi około 14%. Z powodu ograniczeń sprzętowych porównania dokonano jedynie dla dwóch przypadków *A*=2 i *B*=2 oraz *A*=4 i *B*=2.

Tab. 6.4 Model III - porównanie wyników symulacji TCPN i eksperymentów dla średniego czasu odpowiedzi [ms] (średni błąd w weryfikowanych przykładach wynosi 13,9%)

| Obciążenie [zapytań/s] | Rodzaj / Warstwa | Model TCPN | Eksperymenty | Błąd [%] |
|---|---|---|---|---|
| | Model (III) klastra front-end (2 węzły) i back-end (2 węzły) | | | |
| 100 | front-end | 305 | 397 | -30,2 |
| | back-end | 521 | 345 | 33,8 |
| 300 | front-end | 832 | 761 | 8,5 |
| | back-end | 1014 | 801 | 21,0 |
| 500 | front-end | 1131 | 1094 | 3,3 |
| | back-end | 812 | 596 | 24,3 |
| | Model (III) klastra front-end (4 węzły) i back-end (2 węzły) | | | |
| 100 | front-end | 65 | 43 | 33,8 |
| | back-end | 691 | 519 | 24,9 |
| 300 | front-end | 63 | 74 | -17,5 |
| | back-end | 682 | 698 | -2,3 |
| 500 | front-end | 1203 | 983 | 18,3 |
| | back-end | 117 | 108 | 7,7 |

MODEL KLASTRA I REPLIKACJI (MODEL IV)

Dla przypadku replikacji poprawa parametrów w porównaniu do modeli I i II jest bezdyskusyjna. W porównaniu do modelu III wynosi w zależności od przypadku około 0,1[s] dla warstwy. Występuje prawie we wszystkich przypadkach (Tab. 6.5) co świadczy niezbicie o możliwościach osiągnięcia zadawalających wyników w dalszych badaniach.

Tab. 6.5 Model IV - porównanie wyników symulacji TCPN i eksperymentów dla średniego czasu odpowiedzi [ms] (średni błąd w weryfikowanych przykładach wynosi 14,9%)

| Obciążenie [zapytań/s] | Rodzaj / Warstwa | Model TCPN | Eksperymenty | Błąd [%] |
|---|---|---|---|---|
| | Model (IV) klastra front-end (2 węzły) z replikacją back-end (2 węzły) | | | |
| 100 | front-end | 49 | 36 | 26,5 |
| | back-end | 567 | 409 | 27,9 |
| 300 | front-end | 701 | 579 | 17,4 |
| | back-end | 813 | 648 | 20,3 |

| | | | | |
|---|---|---|---|---|
| 500 | front-end | 1001 | 921 | 8,0 |
| | back-end | 1050 | 973 | 7,3 |
| | Model (IV) klastra front-end (4 węzły) z replikacją back-end (2 węzły) | | | |
| 100 | front-end | 75 | 65 | 13,3 |
| | back-end | 1083 | 791 | 26,9 |
| 300 | front-end | 79 | 79 | 0,0 |
| | back-end | 1144 | 910 | 20,4 |
| 500 | front-end | 1042 | 975 | 6,4 |
| | back-end | 965 | 925 | -6,9 |

Największy średni błąd symulacji TCPN i eksperymentów dla modelu IV nie przekracza 15% i związany jest z jego złożonością. Porównanie wyników symulacji dla modeli CSIM i eksperymentów kształtują się podobnie do TCPN. Średni błąd symulacja-eksperymenty wynosi tu odpowiednio: model I - 11,5%, model II - 14,1%, model III - 14,3% i model IV - 16,6%. Tabele 6.6, 6.7, 6.8 i 6.9 porównują średnie czasy odpowiedzi dla poszczególnych modeli.

Tab. 6.6 Model I - porównanie wyników symulacji CSIM i eksperymentów dla średniego czasu odpowiedzi [ms] (średni błąd w weryfikowanych przykładach wynosi 11,5%)

| Obciążenie [zapytań/s] | Rodzaj / Warstwa | Model CSIM | Eksperymenty | Błąd [%] |
|---|---|---|---|---|
| | Model (I) podstawowy | | | |
| 100 | front-end | 10240 | 10756 | -5,0 |
| | back-end | 34 | 29 | 14,7 |
| 300 | front-end | 14852 | 11282 | 24,0 |
| | back-end | 139 | 129 | 7,2 |
| 500 | front-end | 29892 | 27832 | 6,9 |
| | back-end | 302 | 269 | 10,9 |

Tab. 6.7 Model II - porównanie wyników symulacji CSIM i eksperymentów dla średniego czasu odpowiedzi [ms] (średni błąd w weryfikowanych przykładach wynosi 14,1%)

| Obciążenie [zapytań/s] | Rodzaj / Warstwa | Model CSIM | Eksperymenty | Błąd [%] |
|---|---|---|---|---|
| | Model (II) klastra front-end (2 węzły) | | | |
| 100 | front-end | 1 | 1 | 0,0 |
| | back-end | 1 | 1 | 0,0 |
| 300 | front-end | 24 | 22 | 8,3 |
| | back-end | 25 | 15 | 40,0 |
| 500 | front-end | 2484 | 1808 | 27,2 |
| | back-end | 2132 | 1776 | 16,7 |
| | Model (II) klastra front-end (4 węzły) | | | |
| 100 | front-end | 1 | 1 | 0,0 |
| | back-end | 1 | 1 | 0,0 |
| 300 | front-end | 26 | 32 | -23,1 |
| | back-end | 39 | 34 | 12,8 |

| | | | | |
|---|---|---|---|---|
| 500 | front-end | 1003 | 898 | 10,5 |
| | back-end | 1102 | 762 | 30,8 |

Tab. 6.8 Model III - porównanie wyników symulacji CSIM i eksperymentów dla średniego czasu odpowiedzi [ms] (średni błąd w weryfikowanych przykładach wynosi 14,3%)

| Obciążenie [zapytań/s] | Rodzaj / Warstwa | Model CSIM | Eksperymenty | Błąd [%] |
|---|---|---|---|---|
| | Model (III) klastra front-end (2 węzły) i back-end (2 węzły) | | | |
| 100 | front-end | 391 | 397 | -1,5 |
| | back-end | 507 | 345 | 31,9 |
| 300 | front-end | 1056 | 761 | 27,9 |
| | back-end | 1121 | 801 | 28,5 |
| 500 | front-end | 1176 | 1094 | 6,9 |
| | back-end | 513 | 596 | 5,0 |
| | Model (III) klastra front-end (4 węzły) i back-end (2 węzły) | | | |
| 100 | front-end | 51 | 43 | 14,3 |
| | back-end | 88 | 59 | 24,6 |
| 300 | front-end | 59 | 74 | 25,4 |
| | back-end | 73 | 68 | 0,7 |
| 500 | front-end | 110 | 98 | 11,3 |
| | back-end | 131 | 108 | 17,5 |

Tab. 6.9 Model IV - porównanie wyników symulacji CSIM i eksperymentów dla średniego czasu odpowiedzi [ms] (średni błąd w weryfikowanych przykładach wynosi 16,6%)

| Obciążenie [zapytań/s] | Rodzaj / Warstwa | Model CSIM | Eksperymenty | Błąd [%] |
|---|---|---|---|---|
| | Model (IV) klastra front-end (2 węzły) z replikacją back-end (2 węzły) | | | |
| 100 | front-end | 50 | 36 | 28,0 |
| | back-end | 598 | 409 | 31,6 |
| 300 | front-end | 743 | 579 | 22,1 |
| | back-end | 798 | 648 | 18,8 |
| 500 | front-end | 1109 | 921 | 16,9 |
| | back-end | 1083 | 973 | 10,1 |
| | Model (IV) klastra front-end (4 węzły) z replikacją back-end (2 węzły) | | | |
| 100 | front-end | 73 | 65 | 10,9 |
| | back-end | 867 | 791 | 8,8 |
| 300 | front-end | 85 | 79 | 7,1 |
| | back-end | 989 | 910 | 7,9 |
| 500 | front-end | 1191 | 975 | 18,1 |
| | back-end | 950 | 925 | 2,6 |

## 6.5 WNIOSKI Z PORÓWNANIA MODELI I EKSPERYMENTÓW

Utworzono środowisko eksperymentalne, w którego skład wchodzi opracowane przez autora pracy oprogramowanie IGA realizujące podstawowe funkcje giełdy internetowej. Oprogramowanie to przy odpowiednio dobranych parametrach stanowi przykład ISIROSO. W celu umożliwienia porównania opracowanych modeli z wynikami środowiska eksperymentalnego, należało użyć określonego strumienia zgłoszeń SOK. Strumień ten był identyczny jak obciążenie wykorzystywane w modelach. Ponadto należało dostroić czas trwania obsługi.

Eksperymenty referencyjnego modelu giełdy internetowej zakończono na 4 węzłach warstwy *front-end* i 2 węzłach warstwy *back-end* ze względu na ograniczoną ilość sprzętu. Testy rzeczywistego systemu, do tego momentu, w większości przypadków pokrywają się z symulacjami modeli wydajnościowych, dlatego można domniemywać metodą indukcji, że dalsze rozważania teoretyczne są poprawne. Pomimo uproszczeń dokonanych w modelach (Podrozdz. 3.2) i przy podanych założeniach (Podrozdz. 1.2) możliwa jest analiza, która jest odzwierciedleniem modelowanej rzeczywistości. Maksymalny błąd jedynie w dwóch przypadkach, dla średniego czasu odpowiedzi warstwy, przekracza 30%. Dla średnich czasów odpowiedzi warstw oraz długości kolejek, średni błąd jest w granicach 15%. Nie jest to idealny model odwzorowujący modelowaną rzeczywistość, ale pojawiający się błąd w żadnej mierze go nie dyskwalifikuje [73] (Podrozdz. 2.1.1).

Zaletami omawianego podejścia są:

- weryfikacja modeli symulacyjnych,
- zastosowanie analizy ilościowej z TCPN,
- użycie symulatorów wydajności,
- możliwość rozbudowy modeli dzięki użyciu symulatorów,
- możliwość monitorowania elementów systemu w czasie trwania symulacji i eksperymentów.

Do słabych stron proponowanego rozwiązania należą:

- konieczność wytworzenia programów symulujących,
- czasochłonna budowa środowiska laboratoryjnego,
- długie czasy symulacji.

# 7 ZAKOŃCZENIE

Motywacją do podjęcia badań było zetknięcie się z problemami wydajności w eksploatacji rzeczywistych systemów internetowych. W opublikowanych wynikach prac dotyczących projektowania i modelowania systemów internetowych z szybkozmiennymi ofertami dostrzeżony został brak rozwiązań dla tego typu problemów. Autor zauważył również brak dobrych praktycznych rozwiązań. Istniały publikacje, w których modelowano wybrane elementy struktur systemów internetowych, jednak w niewielu z nich podjęto próby modelowania całości architektury systemów, a w żadnej modelowania systemów z szybkozmiennymi ofertami w kontekście wydajności.

Autor w swojej wcześniejszej pracy naukowej zajmował się systemami operacyjnymi [64, 91, 92]. Publikacje te dotyczyły głównie systemu operacyjnego Linux i zastosowań usług sieciowych. Jedna z prac dotyczyła również systemu czasu rzeczywistego RTLinux [62]. Rzeczywiste systemy internetowe zainteresowały autora ze względu na to, że zajmował się wcześniej systemami sieciowymi. Przejście z tej dziedziny do systemów internetowych było więc naturalne, a doświadczenia zdobyte przy okazji zajmowania się systemami czasu rzeczywistego skłaniają do stwierdzenia, że przypominają one rozpatrywane w rozprawie systemy klasy ISIROSO.

W wyniku zdobytych doświadczeń autor postanowił przeprowadzić usystematyzowane badania systemów klasy ISIROSO, objęte pracą doktorską, mające na celu:

- przedstawienie problemu w sposób formalny, dzięki opracowaniu modeli bazowych i ich weryfikację na rzeczywistych systemach,
- przebadanie bardziej złożonych i różnych modeli należących do klasy ISIROSO,
- poszukiwanie rozwiązań architektury cechujących się dobrą skalowalnością,
- przedstawienie wzorców projektowych dla rozważanej klasy systemu,
- określenie praktycznych reguł tworzenia systemów.

W pierwszym etapie utworzony i zweryfikowany został model systemu grupy ISIROSO. Zakończenie tego etapu pozwoliło na opracowanie dobrych podstaw do tworzenia modeli dla złożonych ISIROSO. Wykazano możliwość skutecznego zastosowania narzędzi formalnych (sieci Petriego) i nieformalnych (pakiet CSIM) do wspomagania modelowania i analizy ISIROSO. Zweryfikowane modele pozwoliły na opracowanie zaleceń „jak budować” zrównoważone systemy na bazie wybranych rozwiązań architektury. W wyniku prac opracowano kompleksowe modele umożliwiające analizę jakościową i ilościową systemów klasy ISIROSO.

Postawiona teza rozprawy sprecyzowała zakres badań do jednej z klas systemów internetowych. Tezę zdecydowano się udowodnić poprzez sformułowanie oryginalnej metodyki modelowania bazując na połączeniu metod formalnych i nieformalnych. Wykazano możliwość jednoznacznego wyodrębnienia klasy systemów internetowych z szybkozmiennymi ofertami, dla których czas odpowiedzi wydłuża się ze względu na zwiększającą się liczbę zapytań. Opracowano uniwersalne narzędzia do modelowania, które umożliwiają w szerokim zakresie analizę wydajnościową rozważanych systemów internetowych.

Opracowaną metodykę modelowania zaprezentowano w pięciu głównych rozdziałach, dotyczących odpowiednio:

- Modeli systemów internetowych – W rozdziale 2 zaprezentowano ogólny zarys modelowania i przegląd prac naukowych dotyczących omawianego obszaru. Celem tworzenia modeli formalnych i nieformalnych było porównanie otrzymanych wyników. Rozdział zawiera przegląd możliwych metod modelowania systemów internetowych oraz literatury naukowo-badawczej podejmującej problematykę modelowania wydajności w systemach internetowych. Zaprezentowano używane modele.
- Modeli ISIROSO - Rozdział 3 przedstawia różne systemy e-biznesu. Na podstawie modeli UML omówiono wybraną klasę systemów internetowych z szybkozmiennymi ofertami. Zaprezentowano analityczny opis systemu giełdowego będącego przykładem ISIROSO. Przedstawiono ogólne zasady tworzenia modeli kolejkowego (QN) systemów internetowych. Następnie ideę QN przeniesiono na formalizm TCPN. Tworzenie modelu polegało na konstrukcji sieci TCPN, w której część przejść definiowana jest jako systemy kolejkowe. Zastosowane formalne modele TCPN cechują się wystarczającą „siłą ekspresji" do modelowania i analizy systemów internetowych. Pozwalają one na tworzenie sieci hierarchicznych a czasowe rozszerzenie umożliwia wyrażanie uwarunkowań czasowych. Alternatywnym rozwiązaniem jest bazowanie na typowych modelach symulacyjnych. Do realizacji tej części użyto pakietu symulatora wydajności CSIM.
- Zagadnień doboru architektur dla ISIROSO – W rozdziale 4 metodą krok po kroku przedstawiono sposób, w jaki należy budować złożone modele TCPN i CSIM składające się z klastrów i replikacji, poddać je weryfikacji i użyć do oszacowania wydajności systemów. Modele odzwierciedlają rzeczywistość na pewnym poziomie szczegółowości oraz ukierunkowane są na wybraną klasę systemów internetowych.
- Analizy rozwiązań – W rozdziale 5 utworzone modele zostały poddane analizie w zależności od różnych konfiguracji architektury systemów oraz różnych scenariuszy obciążenia. W celu zrównoważenia systemów przeanalizowano wybrane modele koncentrując się na poprawie wydajności.
- Eksperymentalnej weryfikacji modeli - Rozdział 6 prezentuje weryfikację modeli dokonaną na podstawie wyników z referencyjnego systemu internetowego. Poprzez porównanie czasów odpowiedzi w warstwach modeli symulacyjnych i odpowiadających im konfiguracji systemu referencyjnego, sprawdzono poprawność modeli symulacyjnych. Zastosowanie eksperymentów dało jasną odpowiedź na pytanie dotyczące poprawności modeli symulacyjnych. Weryfikacja modeli przy zastosowaniu rzeczywistego środowiska eksperymentalnego przebiegła pomyślnie.

Zastosowanie warstw oraz replikacji baz danych i klastrów umożliwiło opracowanie sposobów zrównoważenia systemów dla dużych obciążeń. Przeprowadzone symulacje i eksperymenty wykazują przydatność użycia proponowanej metodyki konstrukcji architektury w rzeczywistości.

Oryginalnymi osiągnięciami badawczymi zawartymi w rozprawie wydają się być:

- Utworzenie modeli kolejkowych dla rożnych konfiguracji systemu internetowego. Zastosowanie w modelach klastrów a szczególnie replikacji. Użycie tych modeli jako podstawy do budowy modeli wydajnościowych systemów klasy ISIROSO. Odzwierciedlenie w konstruowanych modelach spotykanego w rzeczywistości wielokrotnego przetwarzania zadań w określonych elementach systemu.
- Opracowanie zasad konstruowania kilku wzajemnie uzupełniających się i coraz bardziej złożonych modeli wydajnościowych internetowego systemu giełdowego (model I, II, III i IV). Użycie klastrów i replikacji do budowy modeli. Zastosowanie klastrów do rozdzielania obciążenia zapytaniami w warstwach. Użycie replikacji pozwoliło na przetwarzanie rozdzielnych transakcji w różnych lokalizacjach bazy danych. Modele te umożliwiły badanie parametrów wydajnościowych systemów klasy ISIROSO, szczególnie czasu odpowiedzi i długości kolejek. Zastosowano więc modelowanie interaktywnego systemu giełdy internetowej pracującego na różnych konfiguracjach architektury i dowolnej liczbie serwerów.
- Zastosowanie formalizmu czasowych kolorowanych sieci Petriego. Utworzenie przy jego użyciu wzorców projektowych systemów kolejkowych FIFO i PS w postaci sieci TCPN. Umieszczenie utworzonych systemów kolejkowych w strukturze sieci w postaci „podstawianych" przejść. Zastosowanie do tego celu narzędzia Design/CPN. Użycie do pełnego opisu modelu funkcji napisanych w języku CPN ML powiązanych z elementami sieci umieszczonymi w węźle deklaracyjnym sieci. Opracowanie zasad konstrukcji i przepływu znaczników w sieci TCPN reprezentujących zapytania, które są przetwarzane w poszczególnych warstwach ISIROSO.
- Utworzenie narzędzia programowego dla utworzonych modeli, na bazie pakietu Design/CPN. Wspomaga ono modelowanie i analizę wydajnościową środowisk internetowych z zastosowaniem czasowych kolorowanych sieci Petriego (TCPN). Narzędzie to w sposób symulacyjny może odwzorowywać czasowe zachowanie sieci kolejkowych i prognozować wydajność w procesie inżynierii wydajności.
- Dzięki użyciu zaimplementowanego w pakiecie Design/CPN modułu analizy wydajnościowej opracowano również efektywne przechwytywanie i analizę danych dla utworzonych modeli. Zastosowanie tego narzędzia umożliwia więc monitorowanie elementów systemu w czasie trwania symulacji.
- Użycie metody, w postaci narzędzia symulacji wydajności CSIM, do modelowania systemów internetowych. Pozwoliło to na badanie charakterystyk wydajności utworzonych modeli systemu klasy ISIROSO przy wykorzystaniu sprawdzonych bibliotek funkcji. Zastosowanie tego pakietu umożliwiło również analizę wyników otrzymanych przy użyciu symulatora bazującego na TCPN. Analizie poddane zostały długości kolejek i czasy odpowiedzi dla identycznych parametrów symulacji w obu przypadkach.
- Porównanie wyników symulacji z wynikami referencyjnego systemu ISIROSO. Implementacja środowiska eksperymentalnego IGA zawierającego mechanizmy klastrowania i replikacji. Utworzenie uproszczonej giełdy internetowej pozwoliło na analizę zrównoważenia rzeczywistego systemu przy różnych konfiguracjach. Dzięki IGA dokonano głównie weryfikacji poprawności proponowanych modeli

oraz weryfikacji stosowanych rozkładów dla przybywania zapytań i czasu obsługi.

- Zaprezentowanie nowatorskiego sposobu badania systemów internetowych, bazującego na modelach TCPN i CSIM pozwalających prognozować wydajność. Sposób ten nie wymaga modyfikacji węzłów oraz kodów źródłowych używanych aplikacji. Po pozytywnej weryfikacji prezentowanych modeli możliwa jest dalsza analiza z wyłączeniem środowiska eksperymentalnego.

W rozprawie zaproponowano zasadę systematycznej rozbudowy modeli. Przedmiotem analizy były więc modele:

- podstawowy (model I),
- klastra w warstwie *front-end* (model II),
- klastra w warstwie *front-end* i klastra w warstwie *back-end* (model III),
- klastra w warstwie *front-end* i klastra z replikacją w warstwie *back-end* (model IV).

Przedstawione modele relatywnie dobrze pozwalają na prognozowanie czasu odpowiedzi. Porównanie wyników modeli TCPN i eksperymentów dało błąd dla poszczególnych modeli: 6,7%, 15%, 13,9% i 14,9%. Porównanie wyników modeli CSIM i eksperymentów dało błąd: 11,5%, 14,1%, 14,3% i 16,6%. Średni błąd pomiędzy samymi czterema modelami TCPN i CSIM, wynosi odpowiednio: 8,5%, 5,5%, 9,6% i 9,5% dla czasu odpowiedzi oraz 12,2%, 8,7%, 15,8% i 16,7% dla średniej długości kolejek. W porównaniu z referencyjnym środowiskiem średni błąd symulacji dla czasu odpowiedzi wynosi dla TCPN 15,1% i CSIM 14,1%. Wyniki otrzymane dla modeli są zbliżone do wyników pomiarów na referencyjnym systemie giełdy. Udowodniono w ten sposób, że pomimo ograniczeń modeli, można je wykorzystać w procesie planowania wydajności dla dużych systemów klasy ISIROSO. Można powiedzieć, że poziom błędu modeli jest podobny ze wskazaniem na CSIM w przypadku średnich czasów odpowiedzi. Nawet dla złożonych modeli, maksymalny błąd modelowania oscylował w granicach 30%. Zatem, mimo wprowadzonych uproszczeń możliwa jest analiza systemów z grupy ISIROSO. Użycie klastrów jest dobrym rozwiązaniem jednak problem skupia się wtedy na bazie danych w warstwie *back-end*. Samo zastosowanie klastrów w warstwie *back-end* (model III) nie rozwiązuje problemu wzrastającego obciążenia, ponieważ prowadzi do wzrostu liczby zadań trafiających do pojedynczej bazy danych, która w takiej sytuacji ulega przeciążeniu. Jak wynika z analiz w przypadku replikacji, czas przetwarzania transakcji i dodatkowy czas poświęcony na synchronizację danych jest mniejszy niż czas obsługi w jednej centralnej bazie danych. Dlatego, obok zastosowania rozdzielania zapytań w procesie napływania, konieczne jest również rozdzielanie niepowiązanych ze sobą transakcji trafiających do bazy danych.

Możliwe jest uzyskanie krótszych czasów odpowiedzi dzięki zastosowaniu kolejnych węzłów w poszczególnych warstwach, co wykazano na podstawie analizy rozwiązań w rozdziale 5. Nie udało się tego zweryfikować eksperymentalne z powodu dysponowania określoną liczbą komputerów, uniemożliwiającą eksperymenty powyżej sześciu węzłów systemu internetowego. Na podstawie dokonanej weryfikacji wykazano poprawność utworzonych modeli dla klasy ISIROSO.

W rozprawie zawarto wyniki pewnego zamkniętego etapu prac badawczych. W czasie realizacji przedstawionych zadań wyłoniły się zagadnienia, które mogą być kontynuowane w przyszłości:

- wprowadzenia do modelowanych systemów kolejnych warstw i węzłów,
- uszczegółowienia sterowania procesem odrzucania zapytań w systemach, uwarunkowane możliwościami przetwarzania systemów w rożnych konfiguracjach architektury i przy różnych parametrach,
- dynamicznego sterowania procesem napływu i odrzucania zapytań klientów oraz wpływu odrzucania zapytań na działanie systemów,
- priorytetowania zapytań, a tym samym określanie poziomu ich obsługi,
- wyznaczenia prawdopodobieństw przepływów zapytań w systemach,
- analizy sytuacji impulsowego obciążenia (ang. *burst*).

Idealnym rozwiązaniem byłoby doprowadzenie do średniego czasu odpowiedzi wszystkich operacji biznesowych poniżej 0,5[s] [49].

Przedstawione w rozprawie interaktywne systemy internetowe realizujące obsługę szybkozmiennych ofert wymagają dalszych i bardziej szczegółowych badań wydajności. Wykonane badania dotyczące wpływu wzrastającego strumienia zapytań na rozważane systemy internetowe potwierdzają słuszność założonego celu pracy. Modelowanie i przeprowadzone eksperymenty wykazują przydatność użycia proponowanej przez autora propozycji dotyczącej modeli i rozwiązań architektonicznych rzeczywistego systemu. Utworzone i zweryfikowane modele wydajności wskazują na słuszność postawionej w pracy tezy.

# BIBLIOGRAFIA


[1] van der Aalst W. M. P.: Interval Timed Coloured Petri Nets and thier Analysis, Application and Theory of Petri Nets, Proceedings of the 14th International Petri Net Conference, Chicago, Lecture Notes in Computer Science vol. 691, Springer-Verlag, pp. 452-467, 1993.

[2] Almeida V., Almeida J., Murta C.: Performance Analysis of a WWW Server, Technical Report 1996-018, Computer Science Department, Boston University and UFMG, Aug. 1996.

[3] Almeida J., Almeida V., Yates D.: Measuring the Behaviour of a World-Wide Web Server, NSF Grant CDA-9529403 and CDA-9623865, 1996.

[4] Amza C., Cox A. L., Zwaenepoel W.: Distributed Versioning: Consistent Replication for Scaling Back-End Databases of Dynamic Content Web Sites, Lecture Notes in Computer Science, Springer-Verlag, vol. 2672, pp. 282-304, 2003.

[5] Andressen D., Yang T.: Multi-processor Scheduling with Client Resources to Improve the Response Time of WWW Applications, University of California - Santa Barbara, 1996.

[6] Aversa L., Bestavros A.: Load Balancing a Cluster of Web Servers, Computer Since Department Boston University, 2000.

[7] Back, R.J., Petre L., Paltor I.P.: Analyzing UML Use Cases as Contracts, In UML: Beyond the Standard. Fort Collins, CO: Springer Verlag, vol. 1723, pp. 518-533, 1999.

[8] Barford P., Crovella M.: Generating Representative Web Workloads for Network and Server Performance Evaluation, pp. 151-160, Madison WI, 1998.

[9] Bause F., Kritzinger F.: Stochastic Petri Nets—An Introduction to the Theory, second ed., Vieweg Verlag, 2002.

[10] Bause F., Buchholz P., Kemper P.: QPN-Tool for the Specification and Analysis of Hierarchically Combined Queueing Petri Nets, Quantitative Evaluation of Computing and Communication Systems, pp. 224-238, Springer-Verlag, 1995.

[11] Bause F., Kemper P.: QPN-Tool for Qualitative and Quantitative Analysis of Queueing Petri Nets, Computer Performance Evaluation, 1994.

[12] Bause F.: QN + PN = QPN - Combining Queueing Networks and Petri Nets, Technical Report no. 461, Dept. of Computer Science, Univ. of Dortmund, Germany, 1993.

[13] Bause F.: Queueing Petri Nets - A Formalism for the Combined Qualitative and Quantitative Analysis of Systems, In Proc. of the 5th International Workshop on Petri Nets and Performance Models, Toulouse (France), 1993.

[14] Bolch G., Greiner S., de Meer H., Trivedi K.S.: Queueing Networks and Markov Chains, Second Edition, Modeling and Performance Evaluation with Computer Science Applications, John Wiley & Sons, 2006.

[15] Bębel B., Wrembel R.: Replikacja danych w bazach danych, 8 Seminarium PLOUG, Warszawa, marzec 2003.

[16] Beckers J., Hendrawan I., Kooij R.E., van der Mei R.: Generalized Processor Sharing Performance Model for Internet Access Lines, 9th IFIP Conference on Performance Modeling and Evaluation of ATM and IP Networks, Budapest, 2001.

[17] Behravanfar R.: Separation of Data and Presentation for the Next Generation Internet Using the Four-Tier Architecture, Technology of Object-Oriented Languages and Systems, 2001.

[18] Bolch, G., Kirschnick, M.: PEPSY-QNS - Performance Evaluation and Prediction System for Queueing NetworkS, Universität Erlangen-Nürnberg, Institut für Mathematische Maschinen und Datenverarbeitung IV, Tech. Report TR-I4-92-21, Oct. 1992.

[19] Wrycza S., Marcinkowski B., Wyrzykowski K.: Język UML 2.0 w modelowaniu systemów informatycznych, Helion, 2006.

[20] Burke P.: The output of a queueing system, Operations Research, 4, pp. 699-704, 1956.

[21]Buyya R.: High Performance Cluster Computing: Architectures and Systems, Prentice Hall PTR, NJ, USA, 1999 i High Performance Cluster Computing: Programming and Applications, Prentice Hall PTR, NJ, USA, 1999.

[22] Cao J., Andersson M., Nyberg C., Kihl M.: Web Server Performance Modeling Using an M/G/1/K*PS Queue, International Conference on Telecommunication, Feb. 2003.

[23] Cardellini V., Casalicchio E., Calajanni M., Yu P. S.: The State of the Art in Locally Distributed Web-server Systems, ACM Computing Surveys, vol. 34, no. 2, pp. 263-311, June 2002.

[24] Cardoso, J, Sibertin-Blanc C.: Ordering action in Sequence Diagrams of UML, in Proceedings of the 23rd International Conference on Information Technology Interfaces, pp. 3-14, 2001.

[25] Chen X., Mohapatra P.: Performance Evaluation of Service Differentiating Internet Servers, IEEE Trans. Computers 51(11), pp. 1368-1375, 2002.

[26] Cheng Z., Singh M. P., Vouk M. A.: Verifying Constraints on Web Service Compositions, International World Wide Web Conference, Proceedings of the 14th International Conference on World Wide Web, Japan, pp. 750-759, 2005.

[27] Chen C., Zhou W.: Towards an Interactive Architecture for Web-Based Databases, Lecture Notes in Computer Science, vol. 3033, pp. 871-878, 2004.

[28] Colajanni M., Yu P. S., Cardellini V.: Scalable Web-Server systems: architectures, models and load balancing algorithms, SIGMETRICS, 2000.

[29] Christensen S., Jørgensen J., Kristensen L.: Design/CPN - A Computer Tool for Coloured Petri Nets, Lecture Notes In Computer Science, Vol. 1217, Proceedings of the 3rd International Workshop on Tools and Algorithms for Construction and Analysis of Systems, pp. 209 - 223, 1997.

[30] Christensen S., Kristensen L. M., Mortensen K. H., Thomasen J. S.: Capacity Planning of Web Servers using Timed Hierarchical Coloured Petri Nets, Proc. of Hewlett-Packard Open View University Association, 6th Plenary Workshop, 1999.

[31] Connoly T., Begg C.: Systemy baz danych, Praktyczne metody projektowania, implementacji i zarządzania, Tom 2, Wydawnictwo RM, Warszawa 2004.

[32] Czachórski T.: A diffusion approximation model of web servers, In Proceedings of IFIP TC6/WG6.4 Workshop on Internet Technologies, Applications and Societal Impact, October 10-11, 2002, Wroclaw, Poland, Cellary W., Iyengar A. (eds. ), Kluwer Academic Publishers, pp. 83-92, Boston, 2002.

[33] Czachórski T.: Modele kolejkowe w ocenie efektywności sieci i systemów komputerowych, PKJS Gliwice 1999.

[34] Czachórski T.: Modele kolejkowe w ocenie efektywności sieci i systemów komputerowych, Wydawnictwo Pracowni Komputerowej Jacka Skalmierskiego, Gliwice, 1999.

[35] Design/CPN Reference Manual for X-Windows, Meta Software Corporation, 1993.

[36] Drakopoulos E.: Performance Modeling and Analysis of a Distributed Computing Environment, 3rd IEEE Symposium on Computers & Communications, 1998.

[37] Edwards H. K., Bauer M. A., Lutfiyya H., Chan Y., Shields M., Woo P.: A Methodology and Implementation for Analytic Modeling in Electronic Commerce Applications, Lecture Notes In Computer Science, vol. 2040, Proceedings of the Second International Symposium on Topics in Electronic Commerce, pp. 148-157, 2001.

[38] Elnikety S., Nahum E., Tracey J., Zwaenepoel W.: A Method for Transparent Admission Control and Request Scheduling in E-Commerce Web Sites, in Proceedings International WWW Conference, New York, USA, 2004.

[39] Filipowicz B.: Modelowanie i optymalizacja systemów kolejkowych, Część I, Systemy markowskie, POLDEX, Kraków 2006.

[40] Filipowicz B.: Modele stochastyczne w badaniach operacyjnych, analiza i synteza systemów obsługi i sieci kolejkowych, WNT, Warszawa 1997.

[41] Gomaa, H.: Object Oriented Analysis and Modeling Families of Systems with UML in 6th International Conference on Software Reuse, Vienna, Austria: Springer Verlag, pp. 89-99, 2000.

[42] Gray J. N., Eswaran K. P., Lorie R. A., Traiger I. L.: The notions of consistency and predicate locks in a database system, Communications of the ACM, 19(11), pp. 624-633, Nov. 1976.

[43] Iyengar A., Nahum E., Shaikh A., Tewari R.: Enhancing Web Performance, IFIP World Computer Congress, 2002.

[44] Jackson J.: Networks of waiting lines, Operational Research, 5, pp. 518-521, 1957.

[45]Jensen K.: Coloured Petri Nets, Basic Concepts, Analysis Methods and Practical Use, vol. 1-3, Springer 1996.

[46] Kamath, M.: Recent Developments in Modeling and Performance Analysis Tools for Manufacturing Systems, In Computer Control of Flexible Manufacturing Systems, (S. Joshi and J. Smith, Eds.), Chapman & Hall, London, pp. 231-263, 1994.

[47] Kang K., Ok M., Park M.: Agent-Based Fair Load Distribution in Linux Web Server Cluster, Lecture Notes in Computer Science, Springer-Verlag, vol. 2918, pp. 143-152, 2004.

[48] Kounev S.: Performance Engineering of Distributed Component-Base Systems, Banchmarking, Modeling and Perrformance Prediction, Shaker Verlag, 2006.

[49] Kounev S.: Performance Modeling and Evaluation of Distributed Component-Based Systems Using Queueing Petri Nets, IEEE Transactions on Software Engineering, vol. 32, no. 7, pp. 486-502, July 2006.

[50] Konunev S., Buchmann A.: Performance Modeling and Evaluation of Large-Scale J2EE Applications, In Proceedings of the 29th Int. Conf. of the Comp. Meas. Group on Res. Manag. and Perf. Eval. of Enterprise Comp. Syst., Dallas, Texas, Dec. 7-12, 2003.

[51] Konunev S., Buchmann A.: Performance Modelling of Distributed E-Business Applications using Queuing Petri Nets, Int. Symposium on Performance Analysis of Systems and Software, IEEE, 2003.

[52] Kounev S., Buchmann A.: Improving Data Access of J2EE Applications by Exploiting Asynchronous Processing and Caching Services, Proc. of the 28th International Conference on Very Large Data Bases, 2002.

[53] Kounev S.: A Capacity Planning Methodology for Distributed E-Commerce Applications, Technical Report, Darmstadt University of Technology, 2001.

[54] Kounev S. D.: Performance Prediction, Sizing and Capacity Planning for Distributed E-Commerce Applications, Technical Report, Darmstadt University of Technology, 2001.

[55] Kounev S. D., Nikolov K.: The Analysis Phase in the Development of E-Commerce Software Systems, TOOLS Eastern Europe, Sofia University, Bulgaria, 1999.

[56] Krishnamurthy B., Willis C.: Analyzing Factors That Influence End-To-End Web Performance, AT&T Labs Research, Proceedings of WWW-9 Conference, Amsterdam, May 2000.

[57] Kristiensen L. M., Christensen S., Jensen K.: The Prectitioner's Guide to Coloured Petri Nets, Int. Journal on Software Tools for Technology Transfer, vol. 2, Springer Verlag, pp. 98-132, 1998.

[58] Lal K., Rak T.: Budowa i tuning klastra komputerowego Mosix-Linux, Pomiary Automatyka Kontrola, nr 1, 72-75, 2005.

[59] Lal K., Rak T.: Linux, a technologie klastrowe, MIKOM-PWN, 2005.

[60] Lal K., Rak T.: Linux. Komendy i polecenia. Praktyczne przykłady, HELION, 2005.

[61] Lal K., Rak T.: Wprowadzenie do użytkowania systemów Unix i Linux, Oficyna Wydawnicza PRz, materiały pomocnicze, Rzeszów 2004.

[62] Lal K., Rak T., Orkisz K.: RTLinux system czasu rzeczywistego, HELION, 2003.

[63] Lal K., Rak T.: Klaster da radę, Vogel Publishing, CHIP SPECIAL, z. Jesień, s. 82-85, 2002.

[64] Lal K., Rak T.: Po prostu własny serwer internetowy, HELION, 2002.

[65] Lazowska E. D., Zohorijan J., Scott Graham G., Servick C. K.: Quantitative System Performance Computer System Analysis Using Queueing Network Models, Printice-Hall, New Jersey 1984.

[66] Linstrom B., Wells L.: Design/CPN Perf. Tool Manual, CPN Group, Univ. of Aarhus, Denmark, 1999.

[67] Liu X., Sha L., Diao Y., Froehlich S., Hellerstein J. L., Parekh S.: Online Response Time Optimization of Apache Web Server, Proc. 11th International Workshop on Quality of Service, 2003.

[68] Manjunath D., Bhaskar D. M., Tahilramani H., Bose S. K., Umesh M.N.: QNAT: A Graphical Tool for the Analysis of Queueing Networks, International Conference on Global Connectivity in Energy, Computer, Communication and Control, pp. 320-323, vol. 2, 1998.

[69] Manley S., Courage M., Seltzer M.: A Self-Scaling and Self-Configuring Benchmark for Web Servers. Network Appliance, Microsoft Corporation, Harvard, 1999.

[70] van der Mei R. D., Hariharan R., Reeser P. K.: Web Server Performance Modeling, Telecommunication Systems, vol. 16, no. 3,4, pp. 361-378, 2001.

[71] Menasce' D. A., Almeida V. A. F., Dowdy L.: Performance by Design, Prentice Hall, 2004.

[72] Menasce' D.A., Almeida V. A. F.: Capacity planning for Web performance, Metrics, models, and methods, Prentice Hall PTR, New Jersey, 2002.

[73] Menasce' D. A., Almeida V. A. F.: Performance of Client/Server Systems, Computer Science, vol. 1769, Springer-Verlag, London, 2000.

[74] Murata T.: Petri Nets: Properties, Analysis and Applications, Proc. of the IEEE, vol. 77, no. 4, pp. 541-580, Apr. 1989.

[75] Ng K. T., Siu Y. M.: The Development of A Betting System on The Internet, The International Conference on Information Technology: Coding and Computing, 2000.

[76] Nicola M., Jarke M.: Performance Modeling of Distributed and Replicated Databases, Revised Survey, IEEE Transactions on Knowledge and Data Engineering, vol. 12, no. 4, pp. 645-672, Jul. 2000.

[77] Pacitti E., Ozsu M. T., Coulon C.: Preventive Multi-master Replication in a Cluster of Autonomous Databases, Int. Conf. on Parallel and Distributed Computing, Austria, 2003.

[78] Pai V.S., Aront M., Banga G., Svendsen M., Druschel P., Zwaenpoel W., Nahum E.: Locality-aware request distribution in cluster-based network servers, Proc. of 8th ACM Conf. On Arch. Support for Progr. Languages, San Jose, 1998.

[79] Petri C. A.: Kommunikation mit Automatem, Schriften des IIM nr 2, Instytut fur Instrumentale mathematik, Bonn, 1962.

[80] Popkov T., Oskotski S.: Queueing Model Based QoS Management Prototype for E-commerce Systems, Proceedings of the 2000 conference of the Centre for Advanced Studies on Collaborative, Canada, 2000.

[81] Puliafito A., Riccobene S., M. Scarpa M.: Modelling of Client-Server Systems, 3rd IEEE International Workshop on Modeling, Analysis, and Simulation of Computer and Telecommunication Systems, 1995.

[82] Rak T.: Tworzenie sieci komputerowej, Ćwiczenia praktyczne, HELION, 2006.

[83] Rak T., Świder K.: Replikacje baz danych w praktyce, Bazy danych: Struktury, algorytmy i metody. Architektura, metody formalne i eksploracja danych, II Konferencja Bazy danych: Aplikacje i Systemy, WKŁ, s. 153-162, 2006.

[84] Rak T., Samolej S.: Zastosowanie kolorowanych sieci Petriego do modelowania czasowych właściwości systemów internetowych, Konferencja: Systemy Czasu Rzeczywistego - kierunki badań i rozwoju, XII Konferencja Systemy Czasu Rzeczywistego, WKiŁ, s. 91-100, 2005.

[85] Rak T.: Workflow models of investment system in Internet, 10 International Modelling School of AMSE-UAPL, Alushta (Crimea), Ukraine, pp. 165-170, Sep., 2005.

[86] Rak T.: Internet System Models, 9 International Modelling School of AMSE-UAPL, Alushta (Crimea), Ukraine, pp. 197-200, Sep., 2004.

[87] Rak T.: Wykorzystanie modeli kolejkowych z zastosowaniem symulatora QNAT do badania zależności czasowych w architekturach systemów internetowych, Współczesne problemy systemów czasu rzeczywistego, SCR, WNT, s. 205-215, 2004.

[88] Rak T.: Zastosowanie modeli MVA do analizy czasów odpowiedzi dla serwisów internetowych, SCR, Materiały pokonferencyjne, s. 81-92, Ustroń, 2003.

[89] Rak T.: Model of Internet Client System Service, Rocznik AGH COMPUTER SCIENCE, AGH Kraków, vol. 5, s. 55-65, 2003.

[90] Rak T.: QoS w systemach klastrowych, Oficyna Wydawnicza PRz, Zeszyty Naukowe PRz, t. 198, z. 23, s. 63-72, 2002.

[91] Rak T.: SuSE Linux 7.2. Czarna księga administratora, HELION, 2002.

[92] Rak T., Zieliński J.: Domowe sieci komputerowe - Ćwiczenia praktyczne, HELION, 2002.

[93] Repenning, A., et al., Using Components for Rapid Distributed Software Development, IEEE Software, 2001

[94] Sacha K.: Projektowanie oprogramowania systemów sterujących, Oficyna Wydawnicza Politechniki Warszawskiej, Warszawa 1996.

[95] Samolej S.: Projektowanie systemów wbudowanych z zastosowaniem czasowych kolorowanych sieci Petrego, rozprawa doktorska pod kierunkiem T. Szmuca, Katedra Automatyki AGH, Kraków, 2004.

[96] Sheldon F., Jerath K., Kwon Y., Baik Y: Case Study: Implementing a Web Based Auction System using UML and Component-Based Programming, 26th International Computer Software and Applications Conference, IEEE Computer Society, London, 2002.

[97] Schwetman H.: CSIM19: A Powerfull Tool for Bilding System Models, Proceedings Winter Simulation Conference, B. A. Peters, J. S. Smith, D. J. Medeiros, and M. W. Rohrer, eds., 2001.

[98] Singhmar N, Apte V., Manjunath D.: A Combined LIFO-Priority Scheme for Overload Control of E-commerce Web Servers, International Infrastructure Survivability Workshop: Overloads, Attacks and Failures: the Trade-off against Time, Portugal, 2004.

[99] Sit Y., Wang C., Lau F.: Cyclone: A High-Performance Cluster-Based Web Server with Socket Cloning, The Journal of Networks, Software Tools and Application, Special Issue on Cluster Computing in the Internet, 2002.

[100] Slothouber L.: A Model of Web Server Performance, Star Nine Technologies Incorporated, 1995.

[101] Smith C.: Performance Engineering, Encyclopedia of Software Eng., J.J. Maciniak, ed., pp. 794-810, John Wiley & Sons, 1994.

[102] de Souza e Silva E., Mutz R.R.: Queueing Networks Solutions and Applications Stochastic Analysis of Computer and Communication Systems, Hideaki Talagi (ed.), pp. 319-399, North Holland, 1990.

[103] Sportak M.: Sieci komputerowe - Księga eksperta, Wydanie II - poprawione i uzupełnione, HELION 2004.

[104] Swain J. J.: Imagine new worlds: Simulation survey, OR/MS Today, pp. 38-51, Feb., 1999.

[105] Szmuc T.: Zaawansowane metody tworzenia oprogramowania systemów czasu rzeczywistego, Wydawnictwa AGH, Kraków, 2001.

[106]Szmuc T.: Modele i metody inżynierii oprogramowania systemów czasu rzeczywistego, Krakowskie Centrum Informatyki Stosowanej, Kraków, 1998.

[107] Szpyrka M.: Projektowanie i analiza czasowych kolorowanych sieci Petriego w oparciu o system DESIGN/CPN, VII Konferencja Systemy Czasu Rzeczywistego, s. 205-214, Kraków, 2000.

[108] Tahilramani Kaur H., Manjunath D., Bose S. K.: The Queueing Network Analysis Tool, 8th IEEE International Symposium on Modeling, Analysis, and Simulation of Computer and Telecommunications Systems, 2000.

[109] Teixeira M. M., Santana M. J., Santana R. H.: Using Adaptive Priority Controls for Service Differentiation in QoS-Enabled Web Servers, International Conference on Computation Science, Kraków 2004.

[110] Trivedi K. S.: Probability and Statistics with Reliability, Queueing and Computer Science Applications. John Wiley & Sons, Inc., second edition, 2002.

[111] Urgaonkar B., Pacifici G., Shenoy P., Spreitzer M., Tantawi A.: An Analytical Model for Multi-tier Internet Service and Its Applications, Proceedings ACM SIGMETRICS International Conference on Measurement and Modeling of Computer Systems, pp. 291-302, 2005.

[112] Verbeek H. M. W., van der Aalst W. M. P.: Woflan 2.0: A Petri Netbased Workflow Diagnosis Tool, In M. Nielsen and D. Simpson, editors, Application and Theory of Petri Nets, vol. 1825 of Lecture Notes in Computer Science, pp. 475-484, Springer-Verlag, 2000.

[113] Walukiewicz S.: Programowanie dyskretne, PWN, Warszawa, 1986

[114] Wells L.: Performance Analysis using Coloured Petri Nets, 10th IEEE International Symposium on Modeling, Analysis, and Simulation of Computer and Telecommunications Systems, 2002.

[115] Wells L., Christensen S., Kristensen L. M., Mortensen K. H.: Simulation based Performance Analysis of Web Servers, Proceedings of the 9th International Workshop on Petri Nets and Performance Models, IDEE Computer Society, pp. 59-68, 2001.

[116] Wells L., Lindstrøm B.: The Design/CPN Performance Tool Manual, Department of Computer Science, University of Aarhus, 1998.

[117] Werewka J., Rak T.: Modele systemów internetowych z dynamicznie zmieniającymi się ofertami, Współczesne problemy systemów czasu rzeczywistego, XI Konferencja Systemy Czasu Rzeczywistego, WNT, s. 147-157, 2004.

[118] Werewka J.: Projektowanie symulacji systemów - symulacja systemów zdarzeń dyskretnych, AGH, Kraków 1989.

[119] Wiesmann M., Schiper A.: Comparison of Database Replication Techniques Based on Total Order Broadcast, IEEE Trans. Knowl. Data Eng. 17(4), pp. 551-566, 2005.

[120] Wiesmann M.: Group Communications and Database Replication - Techniques, Issues and Performance, PhD thesis, Polytechnique de Lausanne, Switzerland, May 2002.

[121] Wiesmann M., Pedone F., Schiper A., Kemme B.: Understanding Replication in Databases and Distributed Systems, Proc. of 20th International Conference on Distributed Computing Systems, pp. 464-474, 2000.

[122] Williams L., Smith C.: Performance and Scalability of Distributed Software Architectures: An SPE Approach, Parallel and Distributed Computing Practices, 2002.

[123] Woodside M., Oufimtsev A., Xu J., Murphy L.: Performance Modeling and Prediction of Enterprise JavaBeans with Layered Queueing Network Templates, Proc. Workshop Specification and Verification of Component-Based Systems, 2005.

[124] Wu H., Kemme B., Maverick V.: Eager Replication for Stateful J2EE Servers, Proc. of the Int. Symposium on Distributed Objects and Applications, Cyprus, 2004.

[125] Zatwarnicki K., Borzemski L.: Rozmyto-neuronowa dystrybucja żądań w klastrowych systemach webowych, Nowe technologie sieci komputerowych, Tom 1, pod red. S. Węgrzyna, L. Znamirowskiego, T. Czachórskiego, S. Kozielskiego, WKŁ, str. 81-90, Warszawa, 2006.

[126] Zatwarnicka A., Zatwarnicki K.: Budowa symulatora serwisu webowego z wykorzystaniem pakietu CSIM, Wyd. Politechniki Opolskiej, Zeszyty Naukowe Politechniki Opolskiej, seria: Informatyka, nr 302/2005, z. 2, s. 107-125, Opole, 2005.

[127] Zatwarnicki K., Zatwarnicka A.: Modele oceny wydajności serwisów internetowych, Współczesne problemy sieci komputerowych, pod red. S. Węgrzyna, P. Pochopienia, T. Czachórskiego, WNT, str. 251-259, Warszawa, 2004.

[128] Zieliński K.: Web Services - Architecture of Information Systems in E-conomy Age, Proc. of International Conference, Wspolczesne kierunki rozwoju elektrotechniki, automatyki, informatyki, elektroniki i telekomunikacji, Kraków, czerwiec 2002.
[129] www.ce.uniroma2.it/~casali/download/MMI/CSIM(9-03-06).pdf - Casaliaccho E.: CSIM.
[130] www.mesquite.com - oprogramowanie firmy Mesquite.
[131] www.daimi.au.dk/designCPN - oprogramowanie Design/CPN.
[132] www.gpw.com.pl - Geiłda Papierów Wartościowych w Warszawie.
[133] wiki.daimi.au.dk/cpntools-help/cpn_ml.wiki?cmd=get&anchor=CPN+ML - CPN ML.
[134] wiki.daimi.au.dk/cpntools/cpntools.wiki - cpntools.
[135] curveexpert.webhop.biz - oprogramowanie CurveExpert.
[136] www.daimi.au.dk/designCPN - opis Design/CPN.
[137] fxtrade.oanda.com/index.shtml - OANDA FXTrade.
[138] www-5.ibm.com/pl/info/faktyliczby.html - IBM Polska.
[139] www2.uwindsor.ca/~hlynka/qbook.html - informacje na temat książek dotyczących sieci kolejkowych.
[140] www.labfit.net - oprogramowanie LAB Fit.
[141] ita.ee.lbl.gov/html/contrib/WorldCup.html - log z serwerów z mistrzostw świata w piłce nożnej Paryż 1998r.
[142] ita.ee.lbl.gov/html/traces.html - logi z serwerów webowych do wykorzystania w badaniach symulacyjnych.
[143] www.linuxvirtualserver.org - projekt Linux Virtual Server.
[144] www.manager-magazin.pl - Manager Magazin.
[145] www.nasdaq.com - NASDAQ on-line.
[146] nsnam.isi.edu/nsnam/index.php/User_Information - symulator NS-2.
[147] www.omnetpp.org - OMNET++ - opis produktu.
[148] www4.informatik.uni-erlangen.de/Projects/PEPSY/en/pepsy.html - oprogramowanie PEPSY-QNS.
[149] www.perfeng.com - oprogramowanie inżynierii wydajności.
[150] www2.uwindsor.ca/~hlynka/qsoft.html - informacje na temat oprogramowanie do modelowania sieci kolejkowych.
[151] www.skarbiec.biz/domy-maklerskie/zagraniczne.htm - zagraniczne giełdy papierów wartościowych.
[152] www.skarbiec.biz/domy-maklerskie/biura.htm - biura maklerskie w Polsce.
[153] www.dvs1.informatik.tu-darmstadt.de/staff/skounev/QPME/index.html - oprogramowanie QPME.
[154] ls4-www.informatik.uni-dortmund.de/QPN/QPN-TOOL_article/article/article.html - oprogramowanie QPN-Tool
[155] www.okstate.edu/cocim/raqs - oprogramowanie RAQS
[156] www.ibm.com, www.microsoft.com, www.mysql.com, www.oracle.com, www.postgresql.org, www.progress.com, www.sybase.com - bazy danych.
[157] www.saxobank.com/?id=958&Lan=EN&Au=1&Grp=5 - SaxoTrader on-line.
[158] www.ipipan.waw.pl/~subieta/artykuly/K.Subieta%20publications.htm - e-biznes.
[159] www.realtimeforex.com/rtf/support/user-guide.forex?lang=eng - Forex on-line.
[160] www.tradingsolutions.com/webhelp/tradingsolutions.html - TradingSolutions on-line.
[161] www.uml.org - OMG Unified Modeling Language.
[162] www.lionbridge.com/lionbridge - WebBench - opis produktu, Ziff Davis Media Benchmarks.
[163] www.mindcraft.com - WebStone - opis produktu.
[164] www.xpresstrade.com - XpressTrade on-line.

# DODATEK

Całość dodatku ze względu na swą obszerność została zamieszczona w wersji elektronicznej na płycie CD dołączonej do rozprawy. Poniżej skrótowo omówiono zawartość poszczególnych podrozdziałów dodatku.

## DODATEK A - WYBRANE SZCZEGÓŁY MODELU TCPN

W celu całościowego zobrazowania modelu TCPN podano wybrane opisy zawierające definicje kolorów i funkcji w języku CPN ML powiązanych z elementami sieci (umieszczanymi w węźle deklaracyjnym sieci): FIFO i PS. Zdefiniowano funkcje niezbędne do badania wydajności.

## DODATEK B - WYBRANE SZCZEGÓŁY MODELU CSIM

W celu przedstawienia modelu CSIM podano wybrane elementy zawierające funkcję główną modelu, wybrane funkcje pomocnicze, funkcje odpowiedzialne za zbieranie wyników oraz pliki nagłówkowe.

## DODATEK C - RAPORT CSIM

Zaprezentowano kompletny „log” wynikowy środowiska symulacyjnego pakietu CSIM, obrazujący parametry, które można otrzymać w wyniku symulacji.

## DODATEK D - SCHEMAT NOTOWAŃ CIĄGŁYCH GIEŁDY

Podrozdział prezentuje pełny schemat działania giełdy z notowaniami ciągłymi. Wybrane elementy użyte do budowy modelu uproszczonej giełdy internetowej (IGA) zaznaczono kolorem.

## DODATEK E - WYBRANE DIAGRAMY MODELU UML

Zaprezentowano wybrane diagramy modelu w języku UML. Są to diagramy: przypadków użycia, sekwencji i wdrożenia. Diagramy przygotowano przy użyciu narzędzia *Posejdon for UML*.

# DODATEK F - WYBRANE SZCZEGÓŁY STANOWISKA EKSPERYMENTALNEGO

Dodatek prezentuje projekt systemu rzeczywistego. Pokazane rysunki prezentują zasadę działanie IGA oraz przedstawiają schemat budowy bazy danych.

# DODATEK G - GIEŁDY WALUTOWE

W rozdziale dokonano analizy pięciu najbardziej ciekawych rozwiązań giełd walutowych on-line: FXTrade, RF2000, SaxoTrader, TradingSolution i Xpresstrade. Wyniki analizy zaprezentowano w tabeli.

# DODATEK H - WYBRANE WYNIKI SYMULACJI TCPN

Dodatek ten prezentuje wybrane wykresy będące wynikiem symulacji modelu TCPN. Ich szczegółowe omówienie znalazło się w rozdziale 6.