KARTA OPISU PRZEDMIOTU - SYLABUS

Nazwa przedmiotu
Przetwarzanie rozproszone

Przedmiot

Kierunek studiów
Informatyka
Studia w zakresie (specjalność)

Poziom studiów
pierwszego stopnia
Forma studiów
niestacjonarne

Rok/semestr
4/7
Profil studiów
ogólnoakademicki
Język oferowanego przedmiotu
polski
Wymagalność
obligatoryjny

Liczba godzin

Wykład	Laboratoria	Inne (np. online)
18	16	
Ćwiczenia	Projekty/seminaria	

Liczba punktów ECTS

4

Wykładowcy

Odpowiedzialny za przedmiot/wykładowca:
Dr inż. Arkadiusz Danilecki
email: arkadiusz.danilecki@cs.put.poznan.pl
tel: (0-61) 665-2964
wydział: Instytut Informatyki
adres: ul. Piotrowo 2, 60-965 Poznań

Odpowiedzialny za przedmiot/wykładowca:
Prof. dr hab. inż. Jerzy Brzeziński
email: jerzy.brzezinski@cs.put.poznan.pl
tel: (0-61) 665-2370
wydział: Instytut Informatyki
adres: ul. Piotrowo 2, 60-965 Poznań

Wymagania wstępne

Student rozpoczynający ten przedmiot powinien posiadać podstawową wiedzę w zakresie algorytmów: oceny ich poprawności i złożoności (czasowej, złożoności średniej oraz w najgorszym przypadku), znać podstawowe zasady programowania strukturalnego i/lub obiektowego. Powinien również znać

podstawowe metody, techniki i narzędzia pomocne w budowie programów. Powinien posiadać także podstawową wiedzę na temat zagadnień związanych z programowaniem wielowątkowym, w szczególności znać problemy związane z wzajemnym wykluczaniem oraz zakleszczaniem procesów i wątków.
Powinien sprawnie posługiwać się możliwościami udostępnianymi przez system operacyjny, przy czym preferowany jest system UNIX/Linux. Powinien również posiadać umiejętność formułowania algorytmów i ich programowania z użyciem języka C lub C++. Powinien posiadać umiejętność rozwiązywania podstawowych problemów związanych z programowaniem w systemie sekwencyjnym, oraz umiejętność pozyskiwania informacji ze wskazanych źródeł, w tym także ze źródeł w języku angielskim.
Powinien również rozumieć konieczność poszerzania swoich kompetencji oraz mieć gotowość do podjęcia współpracy w ramach zespołu. Ponadto w zakresie kompetencji społecznych student musi prezentować takie postawy jak uczciwość, odpowiedzialność, wytrwałość, ciekawość poznawcza, kreatywność, kultura osobista, szacunek dla innych ludzi.

Cel przedmiotu

1. Przekazanie studentom podstawowej wiedzy na temat specyfiki systemów rozproszonych i podstawowych różnic dzielących ich od systemów ściśle powiązanych, budowy systemów rozproszonych, analizy złożoności komunikacyjnej i czasowej algorytmów przy uwzględnieniu specyfiki systemów rozproszonych, analizy poprawności algorytmów rozproszonych
2. Zaznajomienie studentów z trendami rozwoju systemów rozproszonych, podstawowymi problemami pojawiającymi się w trakcie tworzenia systemów rozproszonych i realizacji ich typowych zadań, oraz ich rozwiązaniami
3. Umożliwienie studentom nabycia umiejętności konstrukcji aplikacji rozproszonych, posługiwania się wybranymi narzędziami służącymi do implementacji programów w rozproszonym środowisku
4. Rozwijanie u studentów umiejętności rozwiązywania prostych problemów wymagających rozwiązań specyficznych dla systemów rozproszonych, takich jak wyznaczanie globalnego stanu spójnego
5. Wspomaganie kształtowania u studentów umiejętności pracy zespołowej, właściwych nawyków programistycznych, takich jak dokumentowanie kodu.
6. Kształtowanie umiejętności optymalizacji programów, poprzez dobór odpowiednich narzędzi, algorytmów i metod implementacji

Przedmiotowe efekty uczenia się

Wiedza

ma uporządkowaną, podbudowaną teoretycznie wiedzę ogólną w zakresie algorytmów przetwarzania rozproszonego i ich złożoności (K1st_W4)
ma wiedzę o trendach rozwojowych i najistotniejszych nowych osiągnięciach w informatyce, w szczególności odnośnie systemów rozproszonych (K1st_W5)
zna podstawowe metody, techniki i narzędzia stosowane przy rozwiązywaniu prostych zadań informatycznych z zakresu budowy rozproszonych systemów komputerowych, (K1st_W7)
ma uporządkowaną, podbudowaną teoretycznie podstawową wiedzę w zakresie architektury systemów rozproszonych (K1st_W4)

Umiejętności

potrafi ocenić złożoność obliczeniową algorytmów i problemów z zakresu przetwarzania rozproszonego (K1st_U8)
potrafi, formułując i rozwiązując zadania z przetwarzania rozproszonego, zastosować odpowiednio dobrane metody, w tym metody analityczne, symulacyjne lub eksperymentalne (K1st_U4)
potrafi dokonać krytycznej analizy sposobu funkcjonowania systemów i algorytmów rozproszonych i innych informatycznych rozwiązań technicznych i ocenić te rozwiązania, w tym: potrafi efektywnie uczestniczyć w inspekcji oprogramowania oraz ocenić architekturę oprogramowania z punktu widzenia wymagań pozafunkcjonalnych, ma umiejętność systematycznego przeprowadzania testów funkcjonalnych (K1st_U9)
potrafi zgodnie z zadaną specyfikacją zaprojektować, oraz zrealizować algorytm rozproszony, dobierając język programowania odpowiedni do danego zadania programistycznego oraz używając właściwych metod, technik i narzędzi (K1st_U10)
ma umiejętność formułowania algorytmów rozproszonych i ich implementacji z użyciem przynajmniej jednego z popularnych narzędzi (K1st_U11)
potrafi organizować, współdziałać i pracować w grupie, przyjmując w niej różne role oraz potrafi odpowiednio określić priorytety służące realizacji określonego przez siebie lub innych zadania (K1_U18)

Kompetencje społeczne

rozumie, że w informatyce wiedza i umiejętności bardzo szybko stają się przestarzałe (K1st_K1)
zna przykłady i rozumie przyczyny wadliwie działających rozproszonych systemów informatycznych, które doprowadziły do poważnych strat finansowych, społecznych (K1st_K2)

Metody weryfikacji efektów uczenia się i kryteria oceny

Efekty uczenia się przedstawione wyżej weryfikowane są w następujący sposób:

Efekty kształcenia przedstawione wyżej weryfikowane są w następujący sposób:
Ocena formująca:
a) w zakresie wykładów:
- na podstawie odpowiedzi na pytania dotyczące materiału omówionego na poprzednich wykładach;
b) w zakresie ćwiczeń:
- na podstawie oceny bieżącego postępu realizacji zadań,
Ocena podsumowująca:
Sprawdzanie założonych efektów kształcenia realizowane jest przez:
- ocenę umiejętności związanych z realizacją ćwiczeń laboratoryjnych,
- ocenę sprawozdania przygotowywanego częściowo w trakcie zajęć, a częściowo po ich zakończeniu; ocenę przygotowanego przez zespoły studentów projektu w wybranym przez niego języku programowania (spośród C, C++ lub Java) wykorzystującym MPI, PVM lub inne środowisko, przy czym studenci deklarują wkład każdej osoby w realizację projektu
- ocenę i „obronę” przez studenta sprawozdania z realizacji projektu, ocena ta obejmuje umiejętność pracy w zespole, uzasadnienie wyboru rozwiązania w zależności od jego złożoności i przydatności w określonej architekturze systemu, oraz analizę poprawności i złożoności czasowej i komunikacyjnej

rozwiązania.
- ocenę wiedzy i umiejętności wykazanych na kolokwium zaliczeniowym o charakterze problemowym, obejmującego 4 pytania sprawdzające zagadnienia szczegółowo omawiane w trakcie wykładów. Wymagane jest zdobycie co najmniej 7 punktów na 12 możliwych.
Uzyskiwanie punktów dodatkowych za aktywność podczas zajęć, a szczególnie za:
- omówienia dodatkowych aspektów zagadnienia,
- efektywność zastosowania zdobytej wiedzy podczas rozwiązywania zadanego problemu,
- umiejętność współpracy w ramach zespołu praktycznie realizującego zadanie szczegółowe w laboratorium,
- uwagi związane z udoskonaleniem materiałów dydaktycznych,
- wskazywanie trudności percepcyjnych studentów umożliwiające bieżące doskonalenia procesu dydaktycznego.
- Aktywne uczestnictwo w czasie dyskusji mających na celu rozwiązanie postawionych problemów oraz ich rozwiązania

Treści programowe

W ramach wykładu student zapoznaje się z rzeczywistymi przykładami istniejących systemów rozproszonych, poznaje najważniejsze cechy decydujące o specyfice systemów tego rodzaju, poznaje powody tworzenia tego rodzaju systemów.
W dalszej kolejności przedstawiane są podstawowe pojęcia i definicje związane z tematyką przetwarzania rozproszonego: proces rozproszony i sekwencyjny. Student poznaje model formalny procesu sekwencyjnego i procesu rozproszonego oraz pojęcia związane z wykonaniem procesu i historią wykonania. Poznaje zagadnienia związane z aktywnością procesu, o warunkach uaktywnienia, klasycznych modelach żądań. Dalej przedstawione są pojęcia i formalne definicje kanałów komunikacyjnych, predykatów opisujących stan kanału, operacji komunikacyjnych. Przedstawione są także różnice między komunikacją synchroniczną i asynchroniczną. Wykład przedstawia także różne topologie przetwarzania rozproszonego, właściwości przetwarzania rozproszonego (relacja poprzedzania, diagramy przestrzenno-czasowe, grafy stanów osiągalnych, niedeterminizm przetwarzania,
Następnie student poznaje zagadnienia związane z realizacją czasu logicznego (wirtualnego), oraz poznaje algorytmy Matterna i Lamporta realizujące mechanizmy zegarów wektorowych i skalarnych. Omawiane są także zagadnienia związane z warunkami poprawności algorytmów rozproszonych oraz z analizą złożoności czasowej i komunikacyjnej algorytmów rozproszonych.
Kolejnym zagadnieniem omawianym na wykładzie jest fundamentalne pojęcie spójnego stanu globalnego. Po wprowadzeniu podstawowych definicji (konfiguracji, konfiguracji spójnej, odcięcia i odcięcia spójnego, linii odcięcia) pokazane są możliwe zastosowania algorytmów wyznaczania stanów globalnych (m.in. przedstawione jest pojęcie predykatów globalnych). Wyjaśnione są problemy związane z wyznaczaniem stanu spójnego w systemie w pełni asynchronicznym. Wreszcie omówione są algorytmy konstruujące spójny stan globalny (m.in. algorytm Lamporta dla systemu z kanałami FIFO oraz algorytm Lai-Yanga dla systemów z kanałami non-FIFO).
W dalszej części omawiane są zagadnienia związane z niezawodnością przetwarzania. Zdefiniowane są modele awarii, wprowadzona jest abstrakcja detektora błędów i omówione są różne rodzaje detektorów

błędów oraz przykładowe modele ich realizacji. Pokazane są także sposoby realizacji abstrakcji łączy niezawodnych w oparciu o zawodne fizyczne łącza komunikacyjne oraz omówione są algorytmy realizujące mechanizmy niezawodnej komunikacji grupowej, wykorzystujące detektory błędów o różnych właściwościach.
Wykład prezentuje także problematykę związaną z osiągnięciem konsensusu w systemach rozproszonych. Pokazane jest, dlaczego w ogólności nie jest możliwe wyznaczenie konsensusu w systemie w pełni asynchronicznym, w którym istnieje możliwości awarii chociaż jednego procesu. Następnie przeanalizowane są algorytmy rozwiązujące konsensus przy pewnych dodatkowych założeniach (odnośnie dostępności detektorów błędów określonych klas).
Wykład przedstawia także zagadnienia związane z wyznaczaniem zakończenia w systemach rozproszonych. Wprowadzone są definicje (m.in. zakończenia statycznego i dynamicznego) oraz przedstawione są liczne algorytmy rozwiązujące problem zakończenia dla systemów o różnych topologiach i różnych modelach przetwarzania.
Dla większości przedstawianych algorytmów analizowana jest ich poprawność oraz złożoność czasowa i komunikacyjna.
Na laboratorium studenci zapoznają się z dwoma bibliotekami służącymi do implementacji aplikacji rozproszonych: bibliotekami PVM oraz MPI. Po zapoznaniu się z narzędziami, studenci konfrontują wiadomości uzyskane na wykładzie z praktyczną implementacją algorytmów rozproszonych. Demonstrowane są skutki braku pełnej synchronizacji zegarów nawet w środowisku węzłów połączonych szybką siecią lokalną. Przedstawione są skutki braku synchronizacji między procesami w środowisku rozproszonym. Następnie studenci implementują zegary logiczne, dokonują rozproszenia prostych problemów obliczeniowych (np. łamania haseł metodą przeszukiwania wyczerpującego czy wyliczania wartości π metodą Monte Carlo). Rozwiązywany jest także wybrany klasyczny problem przetwarzania rozproszonego, np. wyznaczania spójnego stanu globalnego lub rozproszonego wzajemnego wykluczania.
Każde laboratorium składa się z przedstawienia problemu, wspólnej dyskusji w celu odnalezienia rozwiązania, a następnie implementacji. W dalszej części, w ramach uzupełnienia wykładu, przedstawione jest wprowadzenie do problemów zakleszczenia oraz wzajemnego wykluczania w kontekście systemów rozproszonych. Studenci następnie pracują w dwuosobowych zespołach. Otrzymują szczegółowe zagadnienia do rozwiązania, wraz z wskazówkami literaturowymi mogącymi pomóc im w zaprojektowaniu rozwiązania. Muszą utworzyć samodzielnie algorytm rozwiązujący rozwiązanie, lub dostosować jeden odnalezionych algorytmów. Przygotowane rozwiązanie musi zostać najpierw zaprojektowane i zaaprobowane, a dopiero po jego analizie zaimplementowane.

Metody dydaktyczne:
1. wykład: prezentacja multimedialna, prezentacja ilustrowana przykładami podawanymi na tablicy, rozwiązywanie zadań, dyskusja
2. ćwiczenia laboratoryjne: rozwiązywanie zadań, ćwiczenia praktyczne, dyskusja, praca w zespole, pokaz multimedialny

Metody dydaktyczne

Wykład: prezentacja multimedialna, ilustrowana przykładami podawanymi na tablicy.
Ćwiczenia laboratoryjne: prezentacja ilustrowana przykładami podawanymi na tablicy oraz wykonanie zadań podanych przez prowadzącego - ćwiczenia praktyczne.

Literatura

Podstawowa

1. Distributed Algorithms, N. Lynch, Morgan Kaufmann Publishers, 1996
2. Ocena stanu globalnego w systemach rozproszonych, J. Brzeziński, Ośrodek Wydawnictw Naukowych, 2001
3. Programowanie współbieżne i rozproszone w przykładach i zadaniach, Z. Weiss, T. Gruźlewski, WNT, 1993
4. Programowanie równoległe i rozproszone, A. Karbowski (red.) E. Niewiadomska-Szynkiewicz (red.), Oficyna Wy-dawnicza Politechniki Warszawskiej, 2009
5. Introduction to Reliable and Secure Distributed Programming, C. Cachin, L. Rodrigues, R. Guerraoui, Springer-Verlag 2011

Uzupełniająca

1. Distributed Algorithms and Protocols, M. Raynal, John Wiley & Sons, 1988
2. Systemy rozproszone: podstawy i projektowanie, G. Coulouris, J. Dollimore, T. Kindberg, Wydawnictwo Naukowo-Techniczne, 1998
3. Distributed Computing: Principles, Algorithms, and Systems, A. D. Kshemkalyani, M. Singhal, Cambridge University Press, 2011
4. Distributed Systems: An Algorithmic Approach, S. Ghosh, Chapman and Hall/CRC 2006
5. Podstawy programowania współbieżnego i rozproszonego, M. Ben-Ari, Wydawnictwo Naukowo Techniczne, 1990
6. Distributed computing. Fundamentals, Simulations and Advanced Topics, Attiya H., Welch J. John Wiley & Sons, 2004

Bilans nakładu pracy przeciętnego studenta

	Godzin	ECTS
Łączny nakład pracy	100	4
Zajęcia wymagające bezpośredniego kontaktu z nauczycielem	35	1,5
Praca własna studenta (studia literaturowe, przygotowanie do zajęć laboratoryjnych/ćwiczeń, przygotowanie do kolokwiów/egzaminu, wykonanie projektu)[1]	65	2,5

1 niepotrzebne skreślić lub dopisać inne czynności