Michał Górniak

[SR] Rozwiązania wybranych zadań z list 1-8

Lista 1 - Zadanie 4. Przezroczystość systemu rozproszonego polega na tym, że mimo tego że system znajduje się na wielu komputerach, to aplikacja sprawia wrażenie, jakby działała na jednym komputerze, na przykład w sensie obcowania użytkownika z tym systemem. Oto wybrane trzy przykłady przezroczystości w systemach rozproszonych:

- przezroczystość współbieżności - ukrycie współbieżnego działania wątków i potencjalnych wyścigów między nimi,
- przezroczystość położenia - ukrycie rozproszenia położenia zasobów, z których dany system korzysta,
- przezroczystość dostępu - ukrycie różnic między sposobami reprezentacji zasobów danych w systemie.

Lista 2 - Zadanie 2. Trzywarstwowa architektura klient-serwer wyróżnia się tym, że architektura aplikacji składa się z następujących trzech warstw:

- warstwa prezentacji - odpowiedzialna za prezentację aplikacji dla użytkownika, to znaczy odpowiada za przygotowanie sposobu na przekazywanie danych użytkownika do aplikacji, a także czytelną prezentację danych wynikających z zapytań wykonanych przez użytkownika,
- warstwa logiki biznesowej - odpowiedzialna za przetwarzanie zapytań pochodzących z warstwy prezentacji, wykonywanie odpowiednich obliczeń oraz przygotowywanie zapytań o dostępie do danych, a następnie ich przetworzenie do oczekiwanej formy,
- warstwa danych - odpowiedzialna za przetwarzanie zapytań o dostępie do danych, przygotowanych przez kod warstwy logiki biznesowej.

Lista 3 - Zadanie 15. Myślę, że odpowiedź nie jest jednoznaczna. Z jednej strony jeżeli serwer nie utrzymuje żadnych dodatkowych informacji o kliencie, to można powiedzieć że jest bezstanowy, z drugiej jednak strony w warstwie transportowej protokołu TCP/IP, która na tym serwerze się znajduje, utrzymujemy dane dotyczące klienta, co mogłoby sugerować, że jest pełnostanowe.

Lista 4 - Zadanie 1. Jednym z powodów jest to, że warstwy są dodawane i usuwane na różnych etapach transportu wiadomości, co ułatwia poprawną obsługę nagłówków na różnych etapach transportu. Warto tutaj też zauważyć, że nagłówki te raczej nie zawierają redundantnych informacji, a jednocześnie dzięki swojej granularności pozwalają na danym etapie transportu wiadomości wybrać dokładnie te informacje, które w tym momencie są potrzebne. W przypadku jednego nagłówka,

musielibyśmy albo go za każdym razem w całości przetwarzać i wyjmować tylko interesujące nas w danym momencie informacje.

Lista 5 - Zadanie 3. Przykłady prawdziwych identyfikatorów:

- numer pesel jako identyfikator dla osób, które taki numer pesel posiadają,
- adres e-mail jako identyfikator skrzynki pocztowej, do której jest przypisany,
- adres MAC jako identyfikator karty sieciowej.

Lista 6 - Zadanie 14. Algorytm zadziała poprawnie. Zauważmy, że jeden procesów, który ogłosi elekcję ma niższy numer id od drugiego, więc ten z większym numerem dostanie wiadomość o elekcji od tego drugiego i na tę wiadomość odpowie, przez co proces o niższym numerze zaprzestanie kontynuowania działania algorytmu tyrana. Dalsza część algorytmu przebiegnie tak samo jak dla jednego procesu ogłaszającego elekcję.

Lista 7 - Zadanie 23. Taki konflikt typu pisanie-pisanie może łatwo powstać na przykład w momencie, gdy dwie instancje klienckie czytają jednocześnie ten sam obiekt danych, który niech będzie liczbą całkowitą, a następnie próbują go na tej podstawie zaktualizować, na przykład dodając do odczytanej wartości pewną inną liczbę całkowitą: pierwsza instancja liczbę A, a druga instancja liczbę B. W ten sposób przy zapisie zapiszemy starą wartość powiększoną o A lub o B, ale na pewno nie o A+B, co byłoby oczekiwanym rezultatem takiego ciągu zdarzeń.

Lista 8 - Zadanie 15. Na początku zamieszczam przywołany w treści rysunek.

Event order	Process P_1	Process P_2	Process P_3	Process P_4
1	sends m_1	receives m_1	receives m_3	sends m_3
2	sends m_2	receives m_3	receives m_1	sends m_4
3		receives m_2	receives m_2	
4		receives m_4	receives m_4	

Figure 8.27: Four processes in the same group with two different senders, and a possible delivery order of messages under FIFO-ordered multicasting.

W przypadku dostarczania uporządkowanego zgodnie z FIFO, możliwe jest każda kombinacja, która zapewnia, że wiadomości wysłane z wybranego procesu, występują w tej kombinacji uporządkowane tak samo, jak zostały wysłane. Natomiast procesy P_2 oraz P_3 mogą mieć różne kombinacje otrzymywania wiadomości, byle tylko kombinacja każdego z nich była jedną z poniższych dozwolonych (dla czytelności uporządkowanych leksykograficznie) kombinacji:

- m_1 -> m_2 -> m_3 -> m_4,
- m_1 -> m_3 -> m_2 -> m_4,
- m_1 -> m_3 -> m_4 -> m_2,
- m_3 -> m_1 -> m_2 -> m_4,

- m_3 -> m_1 -> m_4 -> m_2,
- m_3 -> m_4 -> m_1 -> m_2.

Natomiast w przypadku dostarczania całkowicie uporządkowanego każda z powyższych kombinacji jest możliwa, ale musimy mieć dodatkowo zapewnienie, że proces P_2 i proces P_3 otrzymają wiadomości w tej samej kolejności.