STUDIA INFORMATICA 2001
Volume 22 Number 2 (44)

Przemysław SZMAL
Politechnika Śląska, Instytut Informatyki

WIZUALIZACJA PRZEBIEGU WYKONANIA PROGRAMÓW W SIECI INTERNET

Streszczenie. W artykule są rozpatrywane układy wizualizacji obejmujące rozproszone aplikacje połączone poprzez sieć Internet. Przedstawiono sposoby pozyskiwania danych na drodze śledzenia i próbkowania obserwowanych przebiegów. Omówiono sposoby transmisji danych z wariantowym buforowaniem. Wskazano na możliwość skorzystania z licznika przesłanych pakietów do synchronizacji prezentacji przebiegów komunikujących się z użyciem protokołu TCP.

VISUALIZATION OF DISTRIBUTED PROGRAM EXECUTION IN THE INTERNET

Summary. The paper deals with systems for visualization of distributed programs connected through the Internet. Data can be acquired by program execution tracing and sampling. By transmitting data several buffering variants can be applied. Synchronization of the concurrent processes presentation can be achieved using the TCP transmitted package counter.

1. Wstęp

W ostatnim okresie coraz większe znaczenie zyskują rozproszone aplikacje realizowane na stanowiskach komputerowych komunikujących się za pośrednictwem sieci Internet. Podobnie jak w przypadku aplikacji jednostanowiskowych, wiele uwagi poświęca się zagadnieniom wizualizacji przebiegu wykonania programów składających się na te aplikacje. Od strony technicznej wizualizacja jest realizowana przez *systemy wizualizacji* (SW), współpracujące z obserwowanymi programami w ramach tzw. *układów wizualizacji* (UW).

Systemy wizualizacji mają kilka podstawowych zastosowań:

- mogą stanowić integralną część aplikacji użytkowych, wykorzystywaną przy bieżącej kontroli i przy sterowaniu ich działaniem,
- mogą być użyte jako element wspomagający prace nad tworzeniem programów, ich testowaniem i uruchamianiem,
- mogą służyć celom dydaktycznym – ich zastosowanie ułatwia przedstawienie zasady i sposobu funkcjonowania programów i wykorzystywanych przez nie algorytmów.

Wiele informacji na temat wizualizacji oprogramowania można znaleźć m.in. w zbiorze opracowań [4]. Zamieszczono tam również obszerny spis literatury z tej dziedziny.

Cykl przetwarzania danych związany z ich wizualizacją rozpoczyna się od ich akwizycji. Następnie są one poddawane koniecznym transformacjom i w końcu są prezentowane użytkownikowi w odpowiednio dobranej formie graficznej. W ramach akwizycji ma miejsce pozyskiwanie danych i udostępnianie ich SW. Problemy pozyskiwania danych dominują w przypadku wizualizacji pojedynczych przebiegów. Nietrywialne problemy z przekazywaniem do SW danych pojawiają się w przypadku wizualizacji aplikacji współbieżnych, realizowanych z zaangażowaniem kilku procesorów, gdy zależy nam na wiernym oddaniu zależności czasowych zachodzących między powiązanymi ze sobą zdarzeniami. Temu zagadnieniu jest poświęcona m.in. praca [3]. Problemy z przekazywaniem danych występują też w aplikacjach rozproszonych, gdzie trzeba uwzględnić specyficzne uwarunkowania połączeń sieciowych: dostępne protokoły komunikacyjne, przesunięcia czasowe powstające wskutek transmisji przez łącza o zmiennej konfiguracji i o dynamicznie zmieniającym się obciążeniu. Konieczność uwzględnienia sprzężeń między procesami wprowadza dodatkowe komplikacje. Problemy te znajdują swe odbicie w organizacji SW i stosowanych w nich rozwiązaniach konstrukcyjnych. Część z nich dyskutujemy w dalszych punktach tego opracowania.

2. Wybrane wiadomości o budowie i funkcjonowaniu UW

2.1. Konfiguracja rozproszonego UW

Układ wizualizacyjny (UW) tworzą *aplikacja obserwowana* (AO) i system wizualizacji wraz z elementami pośredniczącymi. Przyjmujemy, że w skład AO wchodzi jedna lub więcej części, z których każda jest przystosowana do pracy na oddzielnym stanowisku komputerowym. W ramach każdej z tych części działa jeden lub większa liczba *procesów* (PO). Części (lub ściślej – ich procesy składowe) komunikują się ze sobą za pośrednictwem sieci Internet. Części AO spełniają w UW funkcję źródeł danych. Przyjmujemy, że dane o przebiegu wykonania poszczególnych części AO są przekazywane przez Internet na jedno stanowisko, gdzie jest zlokalizowana część centralna SW, koordynująca funkcjonowanie

całego SW i odpowiedzialna za transformacje i prezentację danych. Stanowisko to będziemy nazywać *centralnym stanowiskiem wizualizacyjnym* (CSW), a stanowiska goszczące części AO – *stanowiskami obserwowanymi* (SO). W określonych okolicznościach stanowisko wizualizacyjne może być jednym ze stanowisk wykorzystywanych przez AO.

Możliwe są bardziej złożone konfiguracje rozproszonych UW (patrz np. [2]). W szczególności, czynności związane z transformacjami i prezentacją danych mogą być realizowane na kilku podporządkowanych stanowiskach. Niezależnie od tego UW może zawierać kilka lokalnych ośrodków koordynujących przetwarzanie. Dla naszych rozważań nie jest to jednak istotne.

2.2. Wymiana danych w sieci Internet – wybrane zasady

2.2.1. Organizacja połączeń sieciowych

Zgodnie z zasadami obowiązującymi w sieci Internet, stanowiska komputerowe komunikujące się za pośrednictwem sieci są identyfikowane na podstawie unikalnego *adresu IP*, przydzielonego im na stałe lub czasowo.

Każde stanowisko dysponuje pewną, potencjalnie nieograniczoną, liczbą ponumerowanych *portów* komunikacyjnych. Port jest tworem abstrakcyjnym. Poszczególne porty danego stanowiska sieciowego są przydzielane do obsługi strumieni informacji wchodzących lub wychodzących ze stanowiska, przesyłanych tu z jednych, a kierowanych do innych stanowisk. Sieć obsługuje połączenia między wybranymi portami konkretnych par stanowisk. Połączenie jest jednoznacznie identyfikowane przez wskazanie adresu IP i numeru portu dla jednej i drugiej strony połączenia.

W programach, tworzonych w różnych językach, jest wykorzystywana dodatkowa abstrakcja – *gniazdko sieciowe* (ang. *socket*). Gniazdko spełnia w programie rolę nadajnika lub odbiornika strumienia danych przekazywanych lub pobieranych przez program poprzez sieć. Gniazdko jest kojarzone ze wskazanym portem stanowiska komputerowego, na którym jest realizowany program.

Połączenia i transmisje są organizowane z zastosowaniem odpowiedniego protokołu sieciowego. Na poziomie programów aplikacyjnych są stosowane m.in. protokoły TCP (Transmission Control Protocol) i UDP (User Datagram Protocol). Pierwszy przewiduje – m.in. – potwierdzanie otrzymania poszczególnych pakietów danych, składających się na strumień, jak też korekcję stwierdzonych błędów transmisji na drodze powtórzenia transmisji. Czynności te wiążą się z pewnym narzutem czasowym, spowalniającym transmisję. Drugi protokół, UDP, nie przewiduje śledzenia przebiegu transmisji – transmisja jest szybsza, jednak ewentualne błędy pociągają za sobą nieodwracalną utratę przekazywanych w danym momencie porcji danych.

Ustalenie części szczegółów transmisji leży w wyłącznej gestii mechanizmów odpowiedzialnych za zarządzanie pracą sieci. W szczególności, aplikacje korzystające z połączeń sieciowych nie mogą decydować, jaką trasą, tzn. z wykorzystaniem których węzłów pośrednich sieci, będą kierowane poszczególne pakiety danych.

2.2.2. Organizacja transmisji sieciowych na poziomie aplikacji – wybrane zalecenia ogólne

Powiązania informacyjne tworzone w ramach konkretnej aplikacji angażują określone zasoby sieciowe, w szczególności łącza i inne elementy infrastruktury sieciowej, które w jakimś stopniu służą również potrzebom innych aplikacji. W związku z tym w ramach aplikacji traktowanych jako całość, jak też na poziomie ich części składowych rozważa się stosowanie rozwiązań sprzyjających równomiernemu obciążaniu łączy, a zarazem ograniczających ich sumaryczne obciążenie. Temu celowi służy rezygnacja z transmisji dużej liczby niewielkich pakietów danych na rzecz mniejszej liczby większych pakietów. Może to wymagać kumulowania danych w buforach do momentu zgromadzenia wystarczająco dużych ich porcji.

2.3. Charakterystyki czasowe przebiegów w ramach UW

Do istotniejszych wskaźników charakteryzujących funkcjonowanie UW należą *natężenie generacji* i *natężenie prezentacji danych*. W UW rozproszonych dochodzą do nich dwa dalsze, ważne także w innych aplikacjach rozproszonych: *natężenie transmisji danych* i *czas udostępnienia danych*. Omawiamy je w następnych podpunktach.

2.3.1. Natężenie generacji, transmisji i prezentacji danych

Przez *natężenie generacji danych* (NGD) będziemy rozumieć liczbę logicznych porcji danych charakteryzujących obserwowany przebieg generowanych w jednostce czasu w związku z jego realizacją. Tymi porcjami danych mogą być np. ocechowane sygnały zdarzeń istotnych (por. niżej w p. 3.2). *Natężenie prezentacji danych* (NPD) określa liczbę porcji danych prezentowanych obserwatorowi w jednostce czasu. *Natężenie transmisji danych* (NTD) to liczba porcji danych przekazywanych w jednostce czasu między komunikującymi się częściami układu. Natężenia mogą być rozpatrywane jako chwilowe, efektywne, średnie, a także jako potencjalne, tzn. maksymalnie możliwe. W typowym przypadku projekcji realizowanej w trybie bezpośrednim (trybami projekcji zajmujemy się bliżej w p. 4) następuje uzgodnienie efektywnych wartości natężeń, tak że zachodzi zależność:

$$NGD_{ef} = NTD_{ef} = NPD_{ef} \tag{1}$$

W konkretnej sytuacji jedno z natężeń przyjmuje swą wartość potencjalną, a pozostałe się do niej dostosowują w obrębie własnych ograniczeń. Na przykład, wartość NGD_{pot} jest decydująca w przypadku, gdy przebieg obserwowany jest związany z procesem odbywającym się w czasie rzeczywistym z uwarunkowaniami zewnętrznymi. NPD_{pot} dominuje, gdy użytkownik spowalnia tempo prezentacji tak, by dostosować je do swoich możliwości percepcyjnych. Potencjalna wartość NTD_{pot} zależy m.in. od rodzaju łącza, determinującego maksymalną przepustowość, od chwilowego obciążenia łącza, wynikającego z liczby użytkowników, jak też od liczby i rodzaju transmisji danych, które użytkownicy w danym momencie realizują lub usiłują realizować. Zależy też od zastosowanego protokołu.

W sytuacji gdy uzgodnienie natężeń nie jest możliwe, konieczne są zmiany organizacyjne. Wartość NTD_{pot} w pewnych granicach można podnieść, m.in. stosując zalecenia przedstawione w p. 2.2.2 lub dobierając inne łącze. Konfliktowi między wartościami NGD_{pot} i NPD_{pot} w pewnych sytuacjach daje się zaradzić wykonując prezentację w trybie retrospektywnym.

2.3.2. Czas udostępnienia danych przekazywanych przez sieć

Czas udostępnienia danych (CUD) jest to czas, po którym określona porcja danych wysłana za pośrednictwem sieci przez jedną część układu staje się dostępna dla innej części układu. Czas CUD można wyrazić wzorem:

$$CUD = CPE + CTS + CPO \tag{2}$$

W tym wzorze:

- CPE oznacza czas poprzedzający ekspedycję, liczony od chwili wypracowania odpowiedniej porcji danych przez aplikację po stronie nadawczej do chwili przekazania jej do gniazdka nadawczego,
- CTS to czas transmisji sieciowej, liczony od przekazania porcji danych do gniazdka nadawczego do momentu, gdy porcja w stanie gotowym do odczytu pojawi się w gniazdku odbiorczym współpracującej aplikacji lub części aplikacji,
- CPO to czas poprzedzający odczyt, liczony od pojawienia się porcji danych w gniazdku odbiorczym do momentu odczytania jej przez aplikację.

CTS jest czasem, przez który przekazywana porcja danych jest w dyspozycji mechanizmów obsługujących sieć. Na CTS wpływają te same elementy, które kształtują chwilową wartość NTD. W szczególności, gdy w transmisji są wykorzystywane węzły pośredniczące, na CTS wpływa wybór trasy połączenia: poszczególne pakiety mogą wędrować różnymi trasami.

Na czasy CPE i CPO w znacznej części składa się czas pobytu informacji w ewentualnych buforach, wprowadzonych na poziomie aplikacji. Na te składowe CUD można wpływać na poziomie konkretnej aplikacji, odpowiednio dobierając organizację

logiczną przekazywania danych między poszczególnymi częściami systemu, z uwzględnieniem cech przyjętego protokołu transmisyjnego.

3. Akwizycja danych w rozproszonych UW

3.1. Budowa układu akwizycji danych

Układ akwizycji danych sprzęga poszczególne części AO z częścią centralną SW. Stosując kryterium lokalizacji, w układzie akwizycji można umownie wyróżnić *część centralną* i *część czołową*. *Część centralna* układu akwizycji rezyduje na centralnym stanowisku wizualizacyjnym. *Część czołową* tworzą *bloki czołowe* rozmieszczone na poszczególnych SO.

Stosując kryteria funkcjonalne układ można podzielić na dwie warstwy: warstwę przedmiotową, odpowiedzialną za przejęcie danych z obserwowanych przebiegów, i warstwę transmisyjną, zaangażowaną przy przesyłaniu danych między odległymi częściami UW.

3.2. Podstawowe techniki pozyskiwania danych o obserwowanych przebiegach

System wizualizacji współpracujący z obserwowanym programem ma za zadanie prezentować obraz graficzny bieżącego stanu programu i zmian stanu związanych z wykonaniem programu. Przedmiotem wizualizacji są dane przetwarzane przez program, jak też przepływ sterowania.

Pozyskiwanie informacji na temat stanu programu i zmian odbywa się w warstwie przedmiotowej układu. Może się odbywać na drodze *śledzenia* przebiegu (ang. *tracing*) lub też na drodze *próbkowania* (ang. *sampling*).

Śledzenie polega na informowaniu SW o zajściu w programie określonych zdarzeń (ang. *events*), istotnych dla prezentacji. Zdarzeniem może być w szczególności wykonanie w programie pewnej operacji. W celu zapewnienia generowania niezbędnych sygnałów, w programie kierującym obserwowanym przebiegiem są umieszczane wywołania odpowiednich procedur należących do SW. Stanowi to jeden z elementów tzw. instrumentacji programu. Obszerną dyskusję problemów instrumentacji programów dla potrzeb wizualizacji wykonania programów można znaleźć w [5]. W programach wykonywanych w trybie interpretacyjnym sygnały mogą być w stosownych momentach generowane przez interpretator. Sygnał informujący o zdarzeniu jest poddawany tzw. cechowaniu, w ramach którego są dołączane wszystkie konieczne dane na temat zdarzenia i kontekstu, w jakim nastąpiło. W praktyce wstawki instrumentacyjne zapewniają dostarczenie tylko części informacji. Brakujące dane SW uzupełnia sam odwołując się do

obserwowanego programu. W UW rozproszonych za ewentualne uzupełnianie danych jest odpowiedzialna część czołowa układu akwizycji. Z uwagi na opóźnienia wnoszone przez transmisje sieciowe część centralna mogłaby się tym zajmować tylko w przypadku obserwowania przebiegów wolnozmiennych.

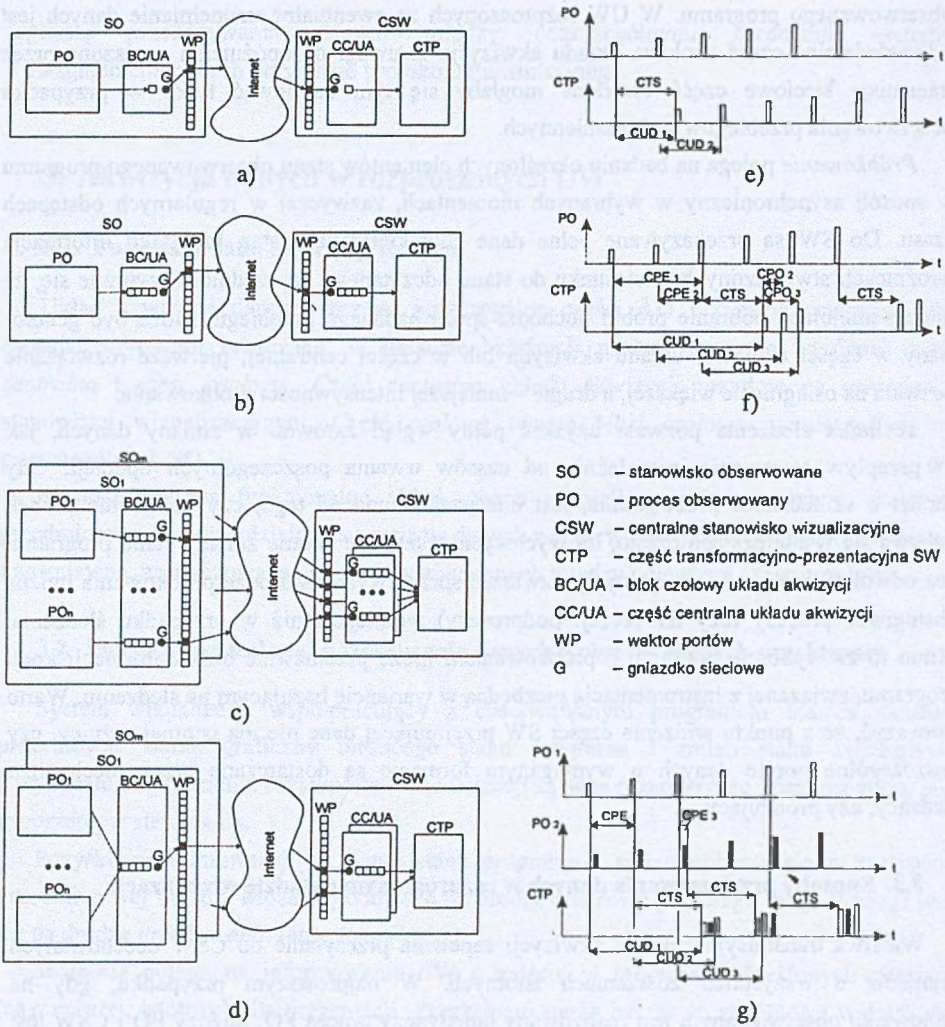
Próbkowanie polega na badaniu określonych elementów stanu obserwowanego programu w sposób asynchroniczny w wybranych momentach, zazwyczaj w regularnych odstępach czasu. Do SW są przekazywane pełne dane charakteryzujące stan lub tylko informacje o różnicach stwierdzonych w stosunku do stanu odczytanego poprzednio. Przyjmuje się, że impuls inicjujący pobranie próbki pochodzi spoza badanego przebiegu. Może być generowany w części czołowej układu akwizycji lub w części centralnej; pierwsze rozwiązanie pozwala na osiągnięcie większej, a drugie – mniejszej intensywności próbkowania.

Technika śledzenia pozwala uzyskać pełny wgląd zarówno w zmiany danych, jak i w przepływ sterowania, niezależnie od czasów trwania poszczególnych operacji. Gdy chodzi o skuteczność próbkowania, jest ona uzależniona od tego, czy pobieranie próbek odbywa się wystarczająco często, by wychwycić wszystkie istotne zmiany stanu programu: bez odwoływania się do specjalnych rozwiązań sprzętowych na drodze próbkowania można obsługiwać procesy (czy też raczej: podprocesy) wolniejsze niż w przypadku śledzenia. Mimo to za wyborem wariantu z próbkowaniem może przemawiać brak deformacji kodu programu, związanej z instrumentacją niezbędną w wariancie bazującym na śledzeniu. Warto zauważyć, że z punktu widzenia części SW przejmującej dane nie ma istotnej różnicy, czy poszczególne porcje danych o wymaganym formacie są dostarczane przez mechanizm śledzący, czy próbkujący.

3.3. Sposoby przekazywania danych w rozproszonym układzie wizualizacji

Warstwa transmisyjna układu akwizycji zapewnia przesyłanie do CSW ocechowanych sygnałów o wszystkich zdarzeniach istotnych. W najprostszym przypadku, gdy na stanowisku obserwowanym jest realizowany pojedynczy proces PO, między PO i CSW jest ustanawiane połączenie, w ramach którego każdorazowo po zaistnieniu zdarzenia wysyłany jest osobny pakiet danych (rys. 1a). O ile porcja danych opisująca zdarzenie nie jest zbyt duża, może być korzystne wprowadzenie jawnego bufora, w którym są gromadzone kolejne sygnały (rys. 1b). Zawartość bufora jest przekazywana, gdy bufor się zapełni, lub wcześniej – po upłynięciu zadanego czasu lub na żądanie CSW.

Użycie bufora powoduje, że zmniejsza się liczba transmitowanych pakietów, tym samym łącze jest lepiej wykorzystywane – w efekcie wzrasta potencjalna wartość natężenia NTD_{pot}. W przypadku gdy łącze jest w stanie przekazać wszystkie pakiety zarówno w wersji buforowanej, jak i niebuforowanej, z równym efektywnym natężeniem transmisji, tj. gdy:



Rys. 1. Sposoby przekazywania danych w rozproszonym UW i związane z nimi czasy udostępniania danych: a) UW z 1 SO i pojedynczym PO, bez jawnego buforowania, b) UW z 1 SO i pojedynczym PO, z jawnym buforowaniem, c) UW z wieloma PO na różnych SO, z jawnymi rozdzielonymi buforami, d) UW z wieloma PO na różnych SO, z jawnym wspólnym buforem, e), f), g) orientacyjne przebiegi czasowe dla konfiguracji a), b) i d)

Fig. 1. Methods for transmitting data in distributed VS and corresponding data availability times: a) VS with 1 OS and single OP, without explicit buffering, b) VS with 1 OS and single OP, with explicit buffering, c) VS with multiple OP's on different OS's with explicit separate buffers, d) VS with multiple OP's on different OS's with common explicit buffer, e), f), g) approximate time diagrams for configurations a), b) and d)

$$NTD^{buf}_{ef} = NTD^{nbuf}_{ef} = NGD \quad (3)$$

czas CUD dla wersji z buforem wydłuża się: przy tej samej wartości składowej CTS wzrastają wartości CPE i CPO (por. rys. 1e-1f). Jednak częściej, gdy $NTD^{buf}_{ef} > NTD^{nbuf}_{ef}$, dla przypadku bez buforowania wzrasta czas CTS i w efekcie czas CUD^{nbuf} może się okazać dłuższy od CUD^{buf}.W przypadku gdy na jednym SO realizowanych jest kilka procesów obserwowanych, można zastosować trzy rozwiązania. Dwa z nich stanowią zwykłe rozszerzenie rozwiązań możliwych w przypadku pojedynczego PO: każdy z PO nawiązuje z CSW osobne połączenie, buforowane (rys. 1c) lub niebuforowane. W trzecim rozwiązaniu jest wykorzystywany wspólny bufor dla sygnałów zdarzeń wszystkich lub części PO realizowanych na jednym SO (rys. 1d). Zaletą tego rozwiązania jest, że wspólny bufor zapełnia się i jest opróżniany szybciej niż bufory indywidualne, zatem średni czas CUD maleje: przy tym samym CTS zmniejsza się suma CPE + CPO (por. zależność (2)). Można to zaobserwować porównując rys. 1f i 1g. Zaleta dodatkowa to angażowanie przy komunikacji mniejszej ilości zasobów komputerów, jakimi są ich porty komunikacyjne.

4. Tryby realizacji prezentacji

Prezentacja może się odbywać w jednym z dwóch trybów: *bieżącym* lub *retrospektywnym*.

W *trybie bieżącym* wizualizowany przebieg i prezentacja odbywają się w tym samym czasie. Dane opisujące przebieg, w miarę jak się pojawiają, są przekazywane do SW, który je sukcesywnie przetwarza i prezentuje obserwatorowi. W zależności od specyfiki wizualizowanego przebiegu tempo projekcji bądź jest dyktowane przez ten przebieg, bądź też jest uzgadniane między programem sterującym tym przebiegiem a SW. Uzgadnianie ma na celu dostosowanie tempa do możliwości percepcyjnych obserwatora.

Jak sygnalizowaliśmy w p. 2.3.1, dyktowanie tempa przez obserwowany przebieg oznacza, że do natężenia generacji danych NGD muszą być dostosowane natężenia NTD i NPD. Wyższe potencjalne wartości NTD można uzyskać stosując buforowanie ograniczające liczbę transmitowanych pakietów danych (por. p. 3.3). W zależności od sposobu wykorzystywania SW w pewnych granicach można też manipulować wartością NPD_{pot}, zwłaszcza gdy czynnikiem limitującym jest percepcja człowieka, a nie moc obliczeniowa stanowiska wizualizacyjnego. Rozpatrzmy UW, w którym istotna jest możliwość interakcji użytkownika z obserwowanymi przebiegami, np. w związku z uruchamianiem aplikacji rozproszonej. Ingerencja ze strony użytkownika następuje, gdy dostrzeże on na urządzeniu wyświetlającym efekt określonego zdarzenia Z. Reakcja systemu (np. w postaci zatrzymania przebiegu) może nastąpić najwcześniej po upłynięciu od chwili zajścia zdarzenia Z czasu CRS:

$$CRS = CUD + CTP + CRU + CUZ \quad (4)$$

gdzie:

- CRS – czas reakcji systemu,
- CUD – czas udostępnienia danych systemowi wizualizacji,
- CTP – czas transformacji i prezentacji danych,
- CRU – czas reakcji użytkownika,
- CUZ – czas udostępnienia PO informacji zwrotnej (liczony analogicznie do CUD).

Należy zwrócić uwagę na składniki CUD i CUZ, efektywne czasy przekazywania danych przez sieć. Czasy te są znikome, gdy komunikujące się stanowiska są zlokalizowane w obrębie tej samej sieci lokalnej. Mogą być zdecydowanie dłuższe, gdy nie należą do tej samej sieci. W pierwszym przypadku można myśleć o pracy w różnych reżimach stosowanych w typowych debuggerach, m.in. o pracy krokowej. W drugim przypadku czas wykonania pojedynczego kroku mógłby być nieakceptowalny, niemniej nadal można by korzystać z predefiniowanych punktów zatrzymań.

Tryb retrospektywny (zwany czasem trybem *post-mortem*) jest stosowany m.in. wtedy, gdy przebieg obserwowany odbywa się w tempie nieodpowiadającym obserwatorowi (za szybko lub zbyt wolno) i nie ma możliwości uzgodnienia tempa. W trybie retrospektywnym obserwowany przebieg i prezentacja są rozdzielone w czasie. Podczas przebiegu jest rejestrowany jego ślad, na który składają się informacje o kolejnych istotnych zdarzeniach, jakie miały miejsce. Podczas prezentacji, odbywającej się w dogodnym dla obserwatora czasie, zarejestrowane w śladzie informacje są odczytywane, przetwarzane i przedstawiane.

Jeżeli wąskim gardłem w układzie są połączenia sieciowe, ślad powinien być rejestrowany w lokalnym otoczeniu obserwowanych procesów, i później przekazywany (być może poprzez sieć) na stanowisko, gdzie odbędzie się prezentacja. Gdy przepustowość sieci jest wystarczająca, dane mogą być rejestrowane centralnie. Warto zauważyć, że w takim przypadku w układzie akwizycji można zastosować te same bloki czołowe co przy prezentacjach realizowanych na bieżąco. Sama rejestracja śladu może się odbywać zamiennie z bieżącą prezentacją lub równolegle z nią.

5. Problemy synchronizacji

Techniki wizualizacyjne są wykorzystywane przy badaniu różnych aspektów wykonania programów. W przypadku aplikacji rozproszonych z reguły występują powiązania i wzajemne uzależnienia czasowe w funkcjonowaniu części składowych aplikacji. Te powiązania nie muszą być przedmiotem zainteresowania obserwatora, a ponieważ stworzenie mechanizmu ich prezentowania jest kłopotliwe, nawet autorzy zaawansowanych sieciowych

systemów animacji algorytmów w pierwszej kolejności rozwiązują inne problemy (por. np. opracowanie [1] dotyczące sieciowego modelu animacji algorytmów *Mocha*).

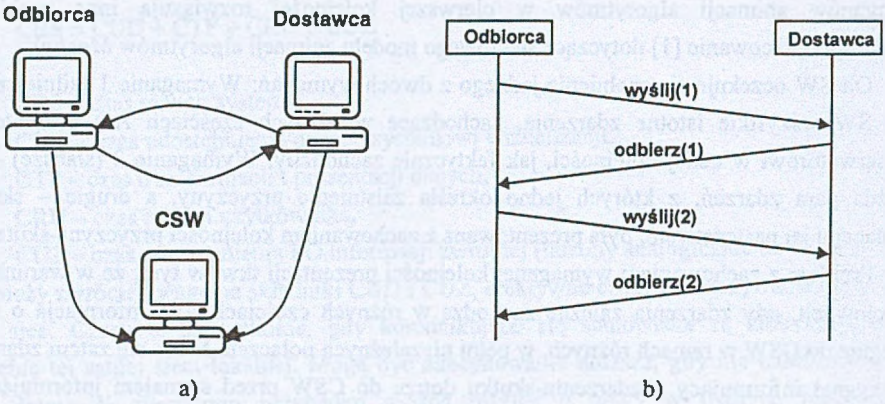
Od SW oczekuje się spełnienia jednego z dwóch wymagań: Wymaganie 1 (silniejsze) – by SW wszystkie istotne zdarzenia, zachodzące w różnych częściach AO prezentował obserwatorowi w takiej kolejności, jak faktycznie zachodziły. Wymaganie 2 (słabsze) – by każda para zdarzeń, z których jedno określa zaistnienie przyczyny, a drugie – skutku będącego jej następstwem, była prezentowana z zachowaniem kolejności przyczyna-skutek.

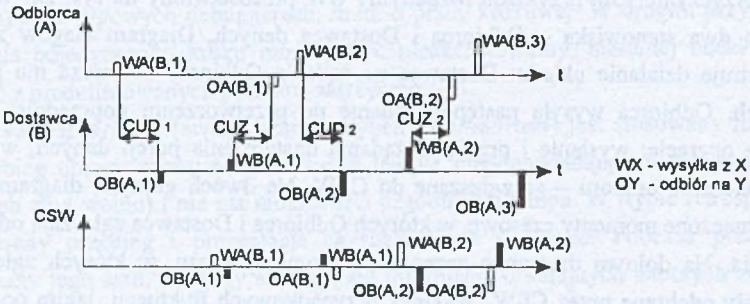
Problem z zachowaniem wymaganej kolejności prezentacji tkwi w tym, że w warunkach sieciowych, gdy zdarzenia zależne zachodzą w różnych częściach AO, informacja o nich biegnie do CSW w ramach różnych, w pełni niezależnych połączeń. Może się zatem zdarzyć, że sygnał informujący o zdarzeniu-skutku dotrze do CSW przed sygnałem informującym o zdarzeniu-przyczynie. Dla przykładu rozpatrzmy UW przedstawiony na rys. 2a. W skład AO wchodzą dwa stanowiska – Odbiorca i Dostawca danych. Diagram ciągów zdarzeń z rys. 2b ilustruje działanie układu: Dostawca na żądanie Odbiorcy dostarcza mu kolejne porcje danych. Odbiorca wysyła następne żądanie po przetworzeniu poprzedniej porcji. Poszczególne operacje: wysłanie i przyjęcie żądania dostarczenia porcji danych, wysłanie porcji i zakończenie odbioru – są zgłaszane do CSW. Na dwóch górnych diagramach na rys. 2c są zaznaczone momenty czasowe, w których Odbiorca i Dostawca zgłaszają odpowiednie zdarzenia. Na dolnym diagramie zaznaczono momenty czasu, w których zgłoszenia zdarzeń zostały odebrane przez CSW. Wskutek przypadkowych fluktuacji, jakim podlegają czasy transmisji danych, w powiązaniu z tym, że transmisje w połączeniach Odbiorca-CSW i Dostawca-CSW nie są koordynowane, odebrana przez CSW sekwencja zdarzeń różni się od faktycznej; sekwencje opisane tekstowo przedstawiają – odpowiednio – rys. 2e i 2d.

Klasyczne rozwiązanie stosowane w celu spełnienia wymagania 1 polega na opatrywaniu sygnałów informujących SW o zdarzeniach *pieczątkami czasowymi* (ang. *time-stamps*). W celu spełnienia wymagania 2 albo korzysta się z pieczątek czasowych, albo stosuje jakąś inną specyficzną formę cechowania sygnałów zdarzeń. Po wpłynięciu do CSW sygnały są poddawane sortowaniu topologicznemu, po czym można je już prezentować obserwatorowi. Procedurę stosuje się zarówno w przypadku prezentacji wykonywanych na bieżąco, jak i w trybie retrospektywnym.

Uzyskanie poprawnych pieczątek czasowych jest trudne, ponieważ wymaga użycia we wszystkich częściach AO synchronicznie pracujących zegarów. W skoncentrowanym systemie wieloprocesorowym opisanym w [3] wykorzystano jeden, centralny zegar systemowy. W warunkach sieci rozległych taki unik nie jest możliwy, również precyzyjna synchronizacja pracy zegarów wydaje się mało realna.

W celu skutecznego cechowania sygnałów trzeba wykorzystać jakiś charakterystyczny element zdarzeń pozostających w związku przyczyna-skutek. Należy zauważyć, że związek





c)

WA(B,1), OB(A,1), WB(A,1), OA(B,1), WA(B,2), OB(A,2), WB(A,2), OA(B,2)

d)

WA(B,1), OB(A,1), WB(A,1), OA(B,1), WA(B,2), OB(A,2), OA(B,2), WB(A,2)

e)

Rys. 2. Przykładowy rozproszony układ wizualizacji: a) struktura układu, b) diagram sekwencji zdarzeń, c) przebiegi czasowe, d) faktyczna sekwencja zdarzeń, e) zaobserwowana sekwencja zdarzeń

Fig. 2. Sample distributed visualization system: a) system structure, b) sequence diagram, c) time diagrams, d) actual sequence of events, e) observed sequence of events

taki powstaje wobec przekazania między różnymi częściami AO konkretnej informacji-sygnału w postaci pewnej porcji danych.

Pierwszy nasuwający się pomysł polega na dołączeniu do danych, przekazywanych w każdym takim momencie, stanu pewnego licznika programowego, zainstalowanego na jednym ze współpracujących stanowisk. Stan tego licznika powinien być podawany SW wraz z sygnałami zgłaszającymi zdarzenie wysłania i odbioru danych. Rozwiązanie

w przedstawionej wyżej postaci nie jest atrakcyjne, gdyż wymagałoby głębokiej ingerencji w budowę i funkcjonowanie AO. Możliwa jest jednak modyfikacja.

Jak wiadomo, niektóre protokoły sieciowe, np. TCP, przewidują zliczanie pakietów danych transmitowanych w ramach danego połączenia. Numer pakietu jest transmitowany wraz z danymi. Stanowi on podstawę do potwierdzenia pomyślnego odbioru poszczególnych pakietów. Do ocechowania sygnałów zdarzeń można wykorzystać właśnie wartość licznika pakietów. Ingerencja w kod aplikacji i jej działanie, niezbędna w celu uzyskania dostępu do wartości licznika, jest zdecydowanie mniejsza niż w przypadku pierwszego, surowego pomysłu. Ilustrację sposobu zastosowania proponowanego rozwiązania można znaleźć na rys. 2. Diagramy czasowe należy tym razem analizować wiedząc, że oznaczenia zdarzeń, wprowadzone na rysunku dla ułatwienia czytelnikowi zorientowania się w rozwoju sytuacji, są w zbliżonej formie faktycznie używane przy zgłaszaniu tych zdarzeń CSW.

Mankamentem proponowanego rozwiązania jest to, że nie może być zastosowane, gdy protokół komunikacyjny nie przewiduje automatycznie obsługiwanego licznika pakietów, co ma miejsce np. w przypadku protokołu UDP.

6. Podsumowanie

W układach wizualizacji przebiegu wykonania aplikacji rozproszonych organizacja układu akwizycji danych istotnie wpływa na własności użytkowe systemu. Jednym z parametrów jest czas udostępniania systemowi wizualizacji informacji o zdarzeniach zachodzących w obserwowanych procesach. Na czas można wpływać stosując różne formy jawnego buforowania przekazywanych w związku z tym danych. W celu poprawnego prezentowania zależności czasowych występujących między zdarzeniami rozgrywającymi się w ramach różnych procesów konieczne jest dostarczenie systemowi wizualizacji dodatkowej informacji umożliwiającej ustalenie czasu, w którym dane zdarzenia następowały. Jako tę informację można wykorzystać stan liczników pakietów danych przesłanych i zaakceptowanych zgodnie z protokołem sieciowym, takim jak np. protokół TCP.

LITERATURA

1. Baker J.E., Cruz I.F., Liotta G., Tamassia R.: Visualizing Geometric Algorithms Over the Web. Brown University Tech. Report CS-97-10, July 1997.
2. Francik J., Szmal P.: Wielostanowiskowa wizualizacja algorytmów w sieciach komputerowych. ZN Pol. Śl. s. Informatyka z. 30, Gliwice 1996, s. 293-310.

3. Kimelman D., Zernik D.: On-the-Fly Topological Sort – A Basis for Interactive Debugging and Live Visualization of Parallel Programs. Workshop on Parallel and Distributed Debugging 1993, s. 12-20.
4. Stasko J., Domingue J., Brown M.H, Price B.A. (red.): Software Visualization. Programming as a Multimedia Experience. The MIT Press, Cambridge, Massachusetts, London, England 1998.
5. Szmal P.: Control Flow Tracing Related to Visualization of Program Execution. Archiwum Informatyki Teoretycznej i Stosowanej, Tom 12 (2000), z. 4, s. 301-322.

Recenzent: Dr inż. Andrzej Białas

Wpłynęło do Redakcji 9 maja 2001 r.

Abstract

In systems for visualization of distributed programs (DPVS), a visualization system (VS) cooperates with some distributed application. We consider configurations where parts of the observed application are executed on several network stations (OS), and the rest of the VS is localised on single central visualization station (Section 2.1). Parts of DPVS communicate according to Internet networking principles and protocols (Section 2.2). Metrics (discussed in Section 2.3) are essential for DPVS functioning: data generation and data presentation intensity, as well as data transmission intensity and data accessibility time. Their influence on possible presentation modes is discussed in Section 4.

DPVS is composed of a data acquisition system and a part in charge of data transformation and presentation. Acquisition system blocks are situated on individual network stations.

Data on interesting events (i.e. requiring some reaction on the VS) can be acquired by program execution tracing and sampling (Section 3.2).

Data transmissions organized on application level (Section 3.3) can use explicit buffering. It helps network links to be used more efficiently. More, in some conditions the time, after which data on events signalled by OS's become available to the transform-presentation part of the system, may decrease. It is worth to use one common buffer for a number of observed processes assigned to the same OS. Configurations of data transmission system completed by approximate time diagrams of transmission related events are shown on Fig. 1.

Synchronization problems are discussed in Section 5. In many VS applications, presentations should preserve actual (absolute), or at least cause-result event order. It is

difficult to use coherent time stamps. However, the events related to application parts communication can be identified using a counter of transmitted data packages; such counter makes a standard part of a package in the TCP protocol. An example of two co-operating processes is used. Illustration for it is Fig. 2.