Paweł Wlaź*

WYKORZYSTANIE TECHNOLOGII VNC DO URUCHAMIANIA PROGRAMÓW W PRZEGLĄDARCE INTERNETOWEJ

Streszczenie. Współczesna technologia komputerowa oferuje różnorodność systemów operacyjnych, środowisk programowania, bibliotek runtime itp. Nie zawsze jest możliwe (lub pożądane) instalowanie programu na komputerze klienckim. Stąd koncepcja uruchomienia takiej aplikacji w oknie przeglądarki WWW. Poza tworzeniem wersji demo (a w niektórych wypadkach nawet w pełni funkcjonalnej aplikacji) gotowej do użycia bez konieczności instalacji, zastosowaniem jest rozszerzenie gamy oprogramowania, które serwer może wykorzystywać na swoich stronach WWW do praktycznie każdego programu dającego się uruchomić na platformie serwera. Po stronie klienta WWW wystarczy przeglądarka obsługująca aplety Javy, po stronie serwera jedyny dodatkowy wymóg to zainstalowanie darmowego serwera VNC. Jest to rozwiązanie tanie, zarówno ze względu na koszty dodatkowego oprogramowania jak i niewielkie wymagania co do zasobów systemu komputerowego.

Słowa kluczowe: Virtual Network Computing, wieloplatformowość, serwery X11.

WSTĘP

Sieć WWW powstała jako technologia zbudowana na bazie protokołu TCP/IP i aplikacji wykorzystujących język HTML, który umożliwia tworzenie tzw. hipertekstu w którym elementy tekstowe i multimedialne przeplatają się z odnośnikami do innych miejsc w sieci. Następnie rozwinięto mnogość technologii wzbogacających możliwości (związane na przykład z interaktywnością, multimedialnością, wykorzystaniem baz danych i.t.p.) WWW. Jednak istnieje i wciąż powstaje dużo oprogramowania typowo desktopowego, o wieloletniej tradycji, wspaniałych możliwościach graficznych, bogatych bibliotekach. Zagadnieniem, któremu zostanie poświęcony ten artykuł, jest uruchomienie takiego programu w przeglądarce WWW, na dowolnej platformie, bez dokonywania w takim programie żadnych zmian, bez ponownej kompilacji, bez tworzenia specjalnej wersji, bez żadnych zmian sposobu wyświetlania i.td.

PROTOKÓŁ REMOTE FRAMEBUFFER ORAZ SYSTEM VIRTUAL NETWORK COMPUTING

Protokół Remote Framebuffer

Przez *Framebuffer* zwykle rozumie się urządzenie kierujące wyświetlaniem na monitorze zawartości bufora pamięci opisującego kompletny układ właściwości pikseli. Zdarza się

* Katedra Matematyki Stosowanej Politechniki Lubelskiej, p.wlaz@pollub.pl

też użycie tego terminu jako samego tylko bufora pamięci (zob. [2,3]). Zatem RFB jest sieciowym protokołem umożliwiającym przesyłanie obrazu przeznaczonego do wyświetlenia na monitorze. Jego specyfikacja jest dostępna w sieci (np.[1]).

RFB powstał w *Olivetti Research Laboratory* w latach 90 jako wsparcie dla komunikacji serwera ze stacjami graficznymi pracującymi w technologii cienkich klientów (*thin clients*). Metoda była oryginalna w stosunku do innych o podobnej funkcjonalności, zaprojektowana od podstaw z myślą o prostocie i minimalnych wymaganiach po obu stronach układu. Bardziej istotne okazało się zastosowanie RFB w opi sanym poniżej systemie VNC. Po zamknięciu *Olivetti Research Laboratory* (2002 r.) praca nad RFB oraz VNC jest kontynuowana, głównie w projektach *RealVNC* [4], *TigerVNC* [5], *TightVNC* [6].

W protokole RFB komunikacja między komputerami odbywa się poprzez protokół TCP. Klient łączy się z serwerem używając portów 5900, 5901, i.t.d. Część implementacji uruchamia również specjalny serwer WWW na porcie o numerze o 100 mniejszym (a więc np. dla serwera uruchomionego na porcie 5910 jest również uruchomiony odpowiadający mu serwer WWW na porcie 5810). Nie ma przeszkód by użyć innych portów TCP, wymaga to jednak modyfikacji standardowego oprogramowania po obu stronach internetowego połączenia.

System Virtual Network Computing

Virtual Network Computing jest systemem, który umożliwia komputerowi-klientowi przechwycenie sesji (np. pulpitu systemu *MS Windows*, sesji serwera w systemie unix/linux, lub sesji Mac OS X). System zasadniczo składa się z oprogramowania dla klienta (oprogramowania umożliwiającego przechwycenie sesji na odległej maszynie) oraz z oprogramowania dla serwera (oprogramowania, które umożliwi „eksport" sesji z danego komputera-serwera do odległego klienta). Podstawowym budulcem umożliwiającym komunikację jest omówiony wcześniej protokół RFB. Ze względu na prostotę przesyłanych danych (informacja o kolorze wypełniającym prostokątne obszary pulpitu) i niezależny od platformy protokół komunikacyjny TCP, system VNC jest z natury wieloplatformowy. Nie ma więc nic dziwnego w obsłudze z komputera z Mac OS X sesji graficznej uruchomionej na FreeBSD, bądź obsłudze pulpitu systemu MS Windows na komputerze z uruchomionym systemem Linux i.t.d. Każda z koncepcji omówionych w dalszym ciągu pracy daje się zrealizować oprogramowaniem otwartym.

Typowe zastosowanie tej technologii jest oczywiste: przechwytywanie pulpitu i klawiatury komputera: czasem w celach serwisu lub pomocy, czasem dla wygodnego sterowania aplikacją uruchomioną na innym komputerze. Często używa się tego rozwiązania do lepszej kontroli serwera pozbawionego fizycznego urządzenia wyświetlającego. Ciekawsze zastosowanie omówię w punkcie wskazującym na różnicę w sesjach VNC dla pulpitu MS Windows i dla sesji serwera X11.

Dla wygody użytkowników, oprócz typowych programów klienckich, w projektach VNC są również obecne aplety javy, które umożliwiają wyświetlenie pulpitu bezpośrednio w przeglądarce. Aplet javy jest oczywiście niezależny od platformy (wymaga jedynie środowiska uruchomieniowego dla wirtualnej maszyny javy, co można uznać obecnie za całkowicie powszechne.

Różnica między serwerem VNC uruchomionym w systemie MS Windows i w X Window

VNC uruchomione w MS Windows wykonuje zadanie zgodne z pierwotnym zamierzeniem: umożliwia przechwycenie innemu komputerowi pulpitu komputera na którym serwer VNC został uruchomiony. To nie jest wystarczające do wykonania zadania o którym mowa w artykule. Nieco inaczej ma się rzecz w przypadku systemu X Window

X Window (zob. [7]) to graficzny system komputerowy ułatwiający programom tworzenie interefejsu komunikującego z użytkownikiem (okienek). System ten składa się z dwu części: serwera, zwanego serwerem X, serwerem X Window, lub serwerem X11 (od głównej cześci numeru wersji systemu, X11 po raz pierwszy opublikowany w 1987 roku) oraz klienta. Serwer to oprogramowanie które na żądanie klienta (np. uruchomionej przez użytkownika aplikacji, bądź menedżera okien) wyświetla elementy graficzne na ekranie, oraz obsługuje urządzenia wejściowe (mysz, klawiatura i.t.p.). Serwer X11 wykonuje jedynie relatywnie proste operacje, nawet zmiana rozmiaru okien jest zadaniem organizowanym przez klienta. Komunikacja między serwerową i kliencka częścią oprogramowania X11 jest zorganizowana sieciowo i w sposób niezależny od fizycznego miejsca pracy tych elementów: klient X11 i serwer X11 mogą funkcjonować w całkowicie różnych lokalizacjach.

X Window jest podstawą systemów okienkowych w wielu odmianach systemu UNIX i Linux. Chociaż MAC OS X (także klasyfikowany jako system uniksowy) ma inny system okienkowy, to można i na nim zainstalować oprogramowanie umożliwiające uruchamianie X11, takie oprogramowanie istnieje także dla MS Windows.

Ważną cechą systemów uniksowych jest to, że ich funkcjonowanie może być całkowicie niezależne od systemu okienkowego, a także to, że uruchomionych serwerów X11 może być w takim systemie więcej niż jeden. Z tego korzysta się przy okazji technologii serwerów VNC. Oczywiście można uruchomić serwer VNC który umożliwi eksportowanie faktycznie używanego pulpitu danej maszyny (czyli powiązanego z odzwierciedlanym fizycznie serwerem X11). Ale jeszcze ciekawsze jest utworzenie wirtualnego serwera X11, który nie jest fizycznie odzwierciedlony w systemie komputerowym serwera, natomiast może być następnie używany za pomocą klienta VNC uruchomionego na dowolnej innej maszynie. To zasadnicza możliwość z której będziemy korzystali w dalszej części pracy.

Na przykład, jeżeli na komputerze z zainstalowanym oprogramowaniem serwera VNC (potrzebuje miejsca rzędu zaledwie kilku MB) wydamy polecenie

```
vncserver :10
```

to zostanie utworzony serwer VNC nasłuchujący na porcie 5910 żądań od klientów VNC. Warto zwrócić uwagę, że taki serwer może zostać uruchomiony przez użytkownika, który nie posiada uprawnień administracyjnych, ponadto takie uruchomienie wymaga jedynie zalogowania się w systemie (na przykład za pomocą *ssh*) i nie jest do tego konieczne jakiekolwiek urządzenie wyświetlające przyłączone do komputera. Jest to więc znakomita opcja dla serwerów fizycznie odseparowanych od świata, do konfiguracji których chcielibyśmy od czasu do czasu używać oprogramowania z graficznym interfejsem użytkownika.

Uruchomiony zostanie nie tylko odpowiednik serwera X11, ale najczęściej także menedżer okien (w systemie uniksowym to menedżer okien zarządza rozmieszczaniem okien, minimalizacją, uruchamianiem aplikacji i.t.d.). Na ogół mamy wiele menedżerów okien do wy-

boru, najpopularniejsze z nich to *kde, gnome, xfce, fluxbox, icewm.* Wyboru dokonuje się w pliku konfiguracyjnym `$HOME/.vnc/xstartup`. Oto przykład zawartości takiego pliku:

```
1.  #!/bin/sh
2.  xsetroot -solid grey
3.  vncconfig -nowin &
4.  fluxbox &
```

Pierwsza linia to specjalny komentarz, wyjaśniający jaki program ma zinterpretować kolejne linie. Druga linia ustawia tło na jednolicie szare (co być może zostanie zmienione przez tło pulpitu menedżera okien włączonego za chwilkę), trzecia linijka sprawi, że system kopiowania/wklejania będzie współdzielony przez oprogramowanie klienta VNC i serwera VNC, czwarta linia uruchamia konkretny program, w tym przypadku menedżer okien *fluxbox,* który następnie zajmie się całą resztą (rysowaniem menu, uruchamianiem aplikacji i.t.d.). Znaki & to zwykłe dla interpretera *sh* komunikaty żądające uruchomienia polecenia w tle i przejścia do następnego.

Uruchomienie takiego wirtualnego serwera jest bardzo wygodną techniką, gdy chcemy by program działał długo. Wtedy wystarczy na działającym bez przerwy serwerze uruchomić taki wirtualny serwer VNC, przyłączyć się do niego klientem VNC, na tym serwerze uruchomić program o który nam chodzi, następnie odłączyć się i można do niego powrócić po kilku dniach. Dotyczy to programów wymagających środowiska graficznego, gdyż dla programów działających w trybie tekstowym, analogiczną funkcjonalność (możliwość uruchomienia na serwerze i wielokrotnego przyłączania/odłączania się) można uzyskać programem *screen*.

URUCHOMIENIE PROGRAMU OKIENKOWEGO DLA PLATFORMY SERWERA W OKNIE PRZEGLĄDARKI KLIENTA

W rozdziale tym opisany zostanie sposób uruchamiania dowolnego programu okienkowego działającego na serwerze, w oknie przeglądarki WWW komputera klienta. To oryginalna metoda wykorzystana z powodzeniem przez autora w kilku przypadkach, używająca technologii opisanych w poprzednich punktach.

Założenia i ogólny schemat połączenia

Będziemy zakładali, że na serwerze uruchomiony jest serwer WWW oraz zainstalowane jest oprogramowanie VNC. Ponadto zainstalowany jest pewien interesujący nas program, skompilowany na platformę serwera, działający w trybie graficznym, którego udostępnienie użytkownikom dowolnej innej platformy nas interesuje. Ze względu na różnice między serwerami, które zostały opisane w punkcie , będziemy zakładali, że na serwerze oprogramowanie graficzne kompilowane jest dla systemu X Window (serwera X11), tak więc dotyczy to wszystkich typowych aplikacji GUI w systemach linux, BSD, Solaris i.t.p.

Naszym zadaniem jest umożliwienie użytkownikowi uruchomienia (i używania) programu, skompilowanego i używającego zasobów serwera, w oknie przeglądarki WWW. Pomysł polega na tym, by w przeglądanej stronie WWW umieścić (np. odpowiednim skryptem CGI lub php, który będzie mógł skorzystać z zasobów serwera):

- żądanie polecenia uruchomiające serwer VNC o odpowiednim, wolnym numerze,
- serwer VNC dla uruchomiającego go użytkownika (np. użytkownika *apache* dla współczesnych wersji popularnego serwera WWW *Apache*) zamiast uruchamiania menedżera okien, po prostu uruchamia interesujący nas program, w pliku konfiguracyjnym można też wpisać dodatkowe elementy konfiguracyjne, jak ścieżki dostępu do programów, by stworzyć naturalne środowisko dla uruchamianego programu, po zakończeniu pracy programu, serwer VNC kończy swoją pracę
- skrypt, którego działanie opisano wyżej musi następnie wygenerować kod uruchamiający aplet javy *vncviewer*, podając mu jako parametry, między innymi, nazwę hosta, numer portu, i inne.

W naszym przykładzie posłużymy się skryptem napisanym w języku *sh* (*Bourne shell*) – a więc popularnym i tradycyjnym na serwerach uniksowych języku skryptowym ogólnego zastosowania. Intuicyjność składni, wygodne odczytywanie wyników pracy innych programów, powszechna dostępność – to ważne zalety, które zadecydowały o wyborze akurat tego języka. Bez problemu można oczywiście osiągnąć podobny efekt np. w języku *php*.

Wykrycie wolnego portu dla serwera VNC

W przykładzie tym, dla naszego programu rezerwujemy 10 serwerów VNC, o numerach od 50 do 59 (czyli porty TCP o numerach od 5950 do 5959). To istotne ustalenie. Musi ono współgrać z liczbą dopuszczalnych, jednoczesnych uruchomień programu, jakie zaplanowaliśmy, z ustawieniami firewalla.

Zatem pętla rozpoczynająca się w linii 3 (zob. ilustracja 1) próbuje znaleźć jeden nieużywany numer VNC z zakresu 50-59. (ten zakres jest skonfigurowany w zmiennej $NUMERY – zob. ilustrację 5). W tym celu uruchamia potok poleceń *netstat* (wyświetlające m.in. połączenia TCP naszego komputera), *fgrep* (wyszukujące wzorzec) i *awk* (w tym wypadku jedynie wypisujący na wyjście szóste wyrażenie z listy podanych) dający jako wynik status połączenia serwera VNC o badanym właśnie numerze. Jeżeli nie ma ono statusu ESTABLISHED (połączenie ustanowione), to oznacza, że ten serwer nie jest wykorzystywany. Oczywiście zakładamy tu, że te numery są zarezerwowane wyłącznie dla naszego celu. W przeciwnym wypadku bowiem, sensowna praca serwera VNC bez chwilowego połączenia z odległym klientem jest czymś najzupełniej normalnym.

Na zakończenie funkcji, wynik jej obliczeń (numer „wolnego” serwera VNC, bądź słowo „brak” jeśli takiego numeru nie ma) zostaje umieszczony w zmiennej `$DOBRY`.

```
1.  function znajdz_numer(){
2.        DOBRY=brak
3.        for NUMER in $NUMERY
4.        do
5.              ES=`netstat -n  | fgrep $SERWER:59$NUMER \
6.                    | tail -1 | awk '{print $6}'`
7.              if [ "$ES" != "ESTABLISHED" ]
8.              then
9.                    let DOBRY=$NUMER
10.             fi
11.       done
12. }
```

Rys. 1. Opis funkcji znajdz_numer()

```
1.  function start_vnc(){
2.        export HOME=/home/demo;
3.        /usr/bin/vncserver -kill :$DOBRY
4.        /usr/bin/vncserver :$DOBRY -SecurityTypes None       \
5.              -NeverShared -AcceptCutText                    \
6.              -SendCutText -DisconnectClients=0              \
7.              -geometry "$SZER"x"$WYS" >& /dev/null &
8.        sleep $OCZEKIWANIE
9.  }
```

Rys. 2. Opis funkcji start_vnc()

Uruchomienie serwera VNC

Gdy już zostanie wykryty wolny numer, kolejną czynnością jest uruchomienie serwera VNC. W naszym przykładowym skrypcie dokonuje tego funkcja `start_vnc()`. Jej pełny opis znajduje się na ilustracji 2. Zostaną teraz omówione jej zasadnicze elementy.

Skrypt CGI jest uruchamiany przez serwer WWW, w tym konkretnym wypadku jest to serwer APACHE, który nie ma swojego typowego „katalogu domowego" (katalogu w którym użytkownik przechowuje m.in. pliki konfiguracyjne uruchamianych aplikacji). Stąd linijka 2 skryptu, która ustawia zmienna środowiskową $HOME na wartość wskazującą „katalog domowy". Jest to niezbędne choćby po to, by *vncserver* wiedział gdzie jest katalog z plikiem x*startup*, który został opisany w punkcie .

Zakładamy, że poprzednio została uruchomiona funkcja *znajdz_numer()*, która ustawiła zmienną $DOBRY na właściwą wartość. To oznacza, że jeden z zarezerwowanych numerów serwera VNC jest wolny, w tym sensie, że nie ma aktywnego połączenia. Jest jednak możliwe, że klient przerwał połączenie (np. zamykając przeglądarkę, ale bez zamykania programu uruchomionego w jej oknie) i choć połączenia nie ma, to serwer na tym numerze wciąż pracuje. Stąd linia 3, która unicestwia serwer VNC działający pod numerem $DOBRY.

Linia 4 (i jej kontynuacje w liniach 5, 6, 7) to uruchomienie serwera o numerze $DOBRY. Polecenie uruchamiające to *vncserver,* jego pierwszy parametr podawany po dwukropku oznacza numer serwera. Parametr `-NeverShared` (linia 5) oznacza żądanie, by do jednego serwera VNC podłączony mógł być co najwyżej jeden klient (to sensowne założenie przy tym rodzaju zastosowania technologii VNC, jakie tu omawiamy). Parametry `-AcceptCutText` (linia 5) oraz `-SendCutText` (linia 6) umożliwiają wymianę danych za pomocą kopiowania/wklejania. Parametr `-DisconectClient=0` (linia 6) zabrania rozłączania istniejącego połączenia w przypadku żądania podłączenia do tego samego serwera VNC. by do jednego serwera VNC podłączony mógł być co najwyżej jeden klient (to sensowne założenie przy tym rodzaju zastosowania technologii VNC, jakie tu omawiamy). Parametr `-geometry` (linia 7) ustawia szerokość i wysokość serwera VNC na podaną liczbę pikseli. Uruchomienie serwera spowoduje jednocześnie – co omówione zostanie w punkcie – uruchomienie właściwej aplikacji. W zależności od mocy serwera i rodzaju aplikacji, może to potrwać nawet kilka sekund, więc żeby w przeglądarce WWW uniknąć pewnych niezręczności związanych z oczekiwaniem na wyświetlenie okna aplikacji, w skrypcie CGI dodajemy linię 8, która spowoduje oczekiwanie z generowaniem dalszej części strony WWW przez liczbę sekund podana w zmiennej $OCZEKIWANIE (zob. także ilustrację 5).

Uruchomienie właściwej aplikacji

W typowym użyciu, plik konfiguracyjny *xstartup* wykorzystywany przez serwer VNC uruchamia menedżera okien, który następnie w zwykły sposób będzie wykorzystywał użytkownik, który się do tego serwera przyłączy. W naszym przypadku postąpimy inaczej. Mianowicie w pliku *xstartup* (w katalogu `$HOME/.vnc`) w przedostatniej linii umieścimy polecenie uruchamiające główną aplikację którą zamierzamy udostępnić w przeglądarce WWW. Ostatnią linią (która zostanie uruchomiona po zakończeniu wykonywana aplikacji) może być zakończenie pracy serwera działającego pod danym numerem.

Takie uproszczone podejście nie zapewni nam pięknych dekoracji okien, ale zminimalizuje zagrożenie przejęcia zasobów komputera przez niepowołana osobę i znacznie uprości przebieg całego procesu.

Wyświetlanie okna *vncservera* za pomocą jako fragmentu strony WWW

Do wyświetlenia *vncservera* uruchomionego opisanymi w poprzednich punktach poleceniami użyjemy apletu Javy. Taki aplet (zwykle w formie archiwum Javy o nazwie *VncVie-*

wer.jar) dostarczany jest przez różne organizacje oferujące oprogramowanie VNC. Różnią się one nazwą głównej klasy i klas pomocniczych (np. w jednym archiwum *jar* do głównej klasy odwołujemy się przez nazwę *VncViewer.class,* w innym *vncviewer/VNCViewer.class*). Poważniejsze różnice dotyczą parametrów przekazywanych apletowi oraz stylowi działania.

W niniejszej pracy użyty został aplet przygotowany przez [6]. Aplet ten przyjmuje szereg parametrów, z których wykorzystamy następujące: *PORT* (to numer portu TCP na którym nasłuchuje serwer VNC do którego aplet ma się podłączyć), *HOST* (to numer IP bądź nazwa domenowa komputera na którym uruchamiony został serwer VNC), *Show Controls* (czy pokazywać panel z elementami sterującymi przyłączeniem do sesji VNC) *Offer Relogin* (czy oferować możliwość ponownego nawiązania zerwanego połączenia). Instrukcje głównego skryptu CGI, który wygeneruje fragment strony odpowiedzialny za uruchomienie apletu zostały ujęte w formie wyodrębnionej funkcji *start_aplet()*. Jej instrukcje zostały wypisane na ilustracji 3.

```
1.  function start_aplet(){
2.  echo '<h1>Drift</h1>'
3.  echo '<applet code="VncViewer.class"    \
4.  echo '       archive='
5.  echo ' "http://mat.pol.lublin.pl/~pwlaz/dryf/VncViewer.jar"'
6.  echo '        width="'$SZER'" height="'$WYS'">'
7.  echo '<param name="PORT" value="59'$DOBRY'" />'
8.  echo '<param name="HOST" value="mat.pol.lublin.pl" />'
9.  echo '<param name="Show Controls" value="No" />'
10. echo '<param name="Offer Relogin" value="No" />'
11. echo '</applet>' }
```

Rys. 3. Funkcja start_aplet()

Linia druga jest oczywiście opcjonalna i nieistotna dla całej koncepcji. w linii trzeciej rozpoczyna się generowanie kodu uruchamiającego aplet (w szczególności podana jest nazwa głównej klasy apletu). W linii piątej podane jest położenie archiwum Javy, które zawiera potrzebne klasy. W linijce 6 podane są (w sposób sparametryzowany) szerokość i wysokość potrzebną do wyświetlenia apletu. Linia 7 podaje numer portu TCP serwera VNC za pomocą obliczonej uprzednio zmiennej $DOBRY, pozostałe linie to nadanie oczywistych wartości omówionym uprzednio parametrom i domknięcie elementu *applet.*

Pozostałe elementy skryptu

Omówione w punktach – fragmenty kodu składają się, jak już to powiedziano, na skrypt CGI, wywoływany na życzenie użytkownika. W skrypcie tym umieszczone są wszystkie omówione już funkcje i jeszcze kilka elementów, które zostaną teraz przedstawione.

Skrypt CGI musi wygenerować pełną stronę w języku opisu dokumentów WWW. Początek tego zadania (wygenerowanie kodu strony aż do elementu otwierającego `<body>`) wykonuje funkcja *start_html()* (ilustracja 4). Strona powinna zostać prawidłowo (w sensie języka opisu stron WWW) domknięta, wykona to funkcja *koniec_html()*. Jej kod jest umieszczony na ilustracji 5.

```
1.  function start_html(){
2.  echo -e "Content-type:text/html\n\n"
3.  echo '<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 '\
4.      'Transitional//EN" "http://www.w3.org/TR/xhtml1'
5.      '/DTD/xhtml1-transitional.dtd">'
6.  echo -e "\n"'<html xmlns="http://www.w3.org/1999/xhtml"'\
7.      ' xml:lang="en" lang="en">'
8.  echo '<head><meta http-equiv="Content-Type"'
9.  echo 'content="text/html; charset=iso-8859-1" />'
10. echo '<link rel="SHORTCUT ICON" href="css/favicon.ico" />'
11. echo '<link rel="StyleSheet"'\
12.     ' href="http://mat.pol.lublin.pl/~pwlaz/dryf/css/'\
13.     'drift.css" />';
14. echo '<link rel="StyleSheet" href="http://mat.pol.'\
15.     'lublin.pl~pwlaz/dryf/css/drift1.css" />'
16. echo '<title>dryf demo</title></head><body>'
17. }
```

Rys. 4. Funkcja start_html() generująca początkową część kodu html

Ilustracja 5 zawiera także pozostałe funkcje skryptu, oraz nadanie wartości istotnym stałym. W liniach 2-3 nadano wartości początkowe stałym używanym przez poprzednio omówione funkcje. Funkcja *akcja()* (linie 10-19) organizuje całą pracę skryptu:

- uruchamia generowanie początkowej części opisu strony,
- wywołuje wyszukiwanie wolnego portu dla serwera VNC,
- jeśli go znajdzie wywołuje uruchomienie serwera VNC na odpowiednim porcie i generowanie kodu związanego z apletem *vncviewer,* w przeciwnym przypadku wywołuje

```
1.  #! /bin/sh
2.  export SZER='850';export WYS='700';export NUMERY=`seq 50 59`
3.  export SERWER='212.182.59.3';export OCZEKIWANIE='2'
4.  function niepowodzenie(){
5.           echo "<p>sorry, you have to try later</p>"
6.  }
7.  function koniec_html(){
8.  echo '</body></html>'
9.  }
10. function akcja() {
11. start_html; znajdz_numer
12. if [ "$DOBRY" = "brak" ]
13. then
14.         niepowodzenie
15. else
16.         start_vnc;start_aplet
17. fi
18. koniec_html
19. }
```

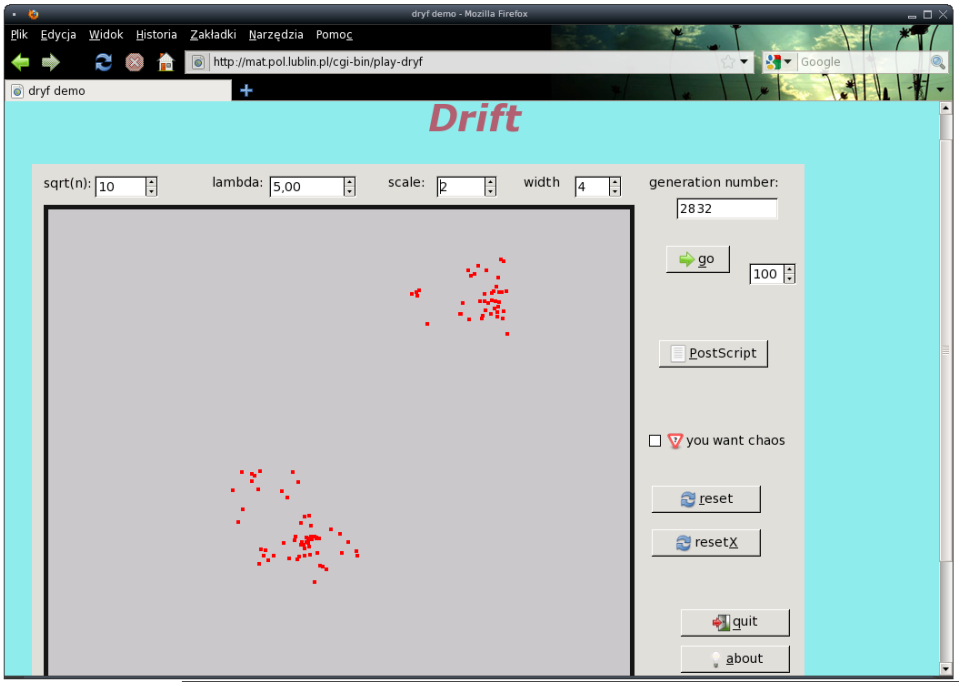
Rys. 5. Pozostałe elementy opisywanego skryptu CGI

funkcję *niepowodzenie()* (która wygeneruje kod wyświetlający komunikat o niemożności uruchomienia aplikacji, linie 4-6),
– na zakończenie wygeneruje kod domykający otwarte elementy *html*.

Pełną zawartość całkowicie funkcjonalnego skryptu CGI uzyskamy doklejając na koniec ilustracji 5 linijki z omówionych ilustracji 1, 2, 3, 4. W ostatniej linii utworzonego pliku umieszczamy polecenie `akcja`, które uruchomi proces wykonujący zadanie naszego skryptu.

PODSUMOWANIE

W pracy została omówiona metoda, która w sposób uniwersalny pozwala dowolny program, także z okienkowym GUI, skompilowany na platformę serwera, uruchomić na serwerze, lecz sterować jego pracą i obserwować wyniki w oknie przeglądarki WWW na komputerze klienta działającym na całkiem dowolnej, współczesnej platformie. Pomysł narodził się przy okazji pewnego programu, który do uruchomienia w systemie Windows wymaga



Rys. 6. Aplikacja działająca na odległym serwerze, „uruchomiona" w oknie przeglądarki

instalowania dodatkowych bibliotek, a więc pewnego skomplikowania systemu dla – być może – jednorazowej potrzeby. Stąd pomysł udostępnienia aplikacji w wersji demonstracyjnej. Omówiona koncepcja umożliwiła to bez konieczności przeprowadzania jakichkolwiek zmian w demonstrowanej aplikacji, która po prostu „działa" w oknie przeglądarki – jak na ilustracji 6.

BIBLIOGRAFIA

1. Richardson T., Levine J.: The Remote Framebuffer Protocol, http://tools.ietf.org/id/draft-levine-rfb-03.txt
2. Shoup R.: SuperPaint: An Early Frame Buffer Graphics System. IEEE Annals of the History of Computing, 2001: 32-37.
3. Protokół RFB, http://en.wikipedia.org/wiki/RFB_protocol#History
4. Real VNC, http://www.realvnc.com
5. Tiger VNC, http://tigervnc.org
6. Tight VNC, http://www.tightvnc.com
7. X Window System, http://pl.wikipedia.org/wiki/X_Window_System

USING VNC TECHNOLOGY FOR RUNNING PROGRAMS IN WWW BROWSERS

Summary

Nowadays computer technology offers variety of operating systems, programming environments, runtime libraries, etc. Sometimes it is not possible (or desired) to install given program on a client computer. Thus the concept of running given application in the window of WWW browser. Apart from making demo versions (or even fully functional applications in some cases) which is ready to use without the need for installation, we also broaden the spectrum of programs that can be used by a server on its WWW pages to virtually all programs that can be run of this server platform. On a client side it is enough to have a browser which is Java applet ready. On a server side the only additional demand is installation of free VNC server. The costs of this solution are very low, in respect to both additional software costs and demands for system resources.

Key words: Virtual Network Computing, cross-platform software, X11 servers.