STUDIA INFORMATICA 2007
Volume 28 Number 1 (70)

Piotr STERA
Politechnika Śląska, Instytut Informatyki

PROTOKOŁY ZDALNEGO DOSTĘPU DO GRAFICZNEGO INTERFEJSU UŻYTKOWNIKA

Streszczenie. Niniejszy artykuł przedstawia protokoły stosowane do tzw. zdalnego dostępu. Koncentruje się on na zagadnieniu ilości danych przez nie przesyłanych. W pierwszej części opisano protokoły zdalnego dostępu, zaś w drugiej przeanalizowano zagadnienie ilości danych przez nie transmitowanych. Następnie porównano rozważania teoretyczne z wynikami testów przeprowadzonych na najczęściej stosowanych aplikacjach tych protokołów oraz podano wnioski wynikające z takiego porównania.

Słowa kluczowe: zdalny dostęp, graficzny interfejs użytkownika, protokół RFB, system X Window, Linux Terminal Server Project, protokół SSH, technologia NX

REMOTE ACCESS PROTOCOLS FOR GRAPHICAL USER INTERFACE

Summary. The paper presents remote access protocols. It focus on the volume of data transmission. First part describes chosen protocols. Second one concentrates on analysis of the amount of transmitted data. Next, theoretic arguments are compared with application tests. Results of comparison are presented as conclusions.

Keywords: remote access, graphical user interface, RFB protocol, X Window system, Linux Terminal Server Project, SSH protocol, NX technology.

1. Wprowadzenie

W tej części artykułu zostały krótko opisane wybrane protokoły zdalnego dostępu do graficznego interfejsu użytkownika.

1.1. System X Window

System X Window został opracowany przez MIT w 1984 roku, rozwojem projektu kieruje organizacja X.Org [1]. Obecnie stosowaną wersją systemu jest X11. System X Window dostarcza standard narzędzi oraz protokół do budowy graficznego interfejsu użytkownika (GUI) [2]. Udostępnione funkcje pozwalają na utworzenie szkieletu GUI – rysowanie i przemieszczanie okien na ekranie oraz obsługę urządzeń wejściowych, takich jak: klawiatura i myszka. W zakresie funkcji graficznych system X Window umożliwia tworzenie na ekranie tzw. prymitywów graficznych, czyli najprostszych elementów (punkty, odcinki, wielokąty, elipsy czy wyświetlanie bitmap), nie zawiera natomiast opisu szczegółów interfejsu graficznego, takich jak: wygląd przycisków czy menu. Do obsługi tych ostatnich przewidziany jest osobny program zwany menadżerem okien (obecnie do najbardziej znanych zalicza się GNOME i KDE).

Ważną cechą systemu jest jego sieciowa przeźroczystość – nie jest istotne, czy poszczególne elementy systemu są uruchomione na jednym komputerze, czy też pomiędzy nimi znajduje się sieć komputerowa. Właściwość ta wynika ze struktury opartej na modelu klient--serwer. Serwer stanowi program wyświetlający ekran na lokalnej stacji użytkownika, zaś klientami są programy, które korzystają z usługi wyświetlania – mogą one działać zarówno na stacji lokalnej, jak i na komputerze komunikującym się przez sieć (który może mieć inną architekturę i system operacyjny). Dzięki takiej budowie systemu X Window zapewniono rozdzielenie logiki aplikacji od jej prezentacji [3], co pozwala uzyskać na jednym ekranie okna programów działających na różnych komputerach i pod kontrolą różnych systemów operacyjnych. [4]. Do najważniejszych wad systemu zalicza się:

- wymaganie od komputera wyświetlającego uruchomienia serwera X, który jest obszernym programem, co może ograniczać możliwości użycia go na niektórych typach komputerów, np. PDA;
- zagrożenie spowodowane zezwoleniem odległemu komputerowi na użycie pulpitu;
- aplikacje uruchamiają się bardzo wolno na łączach o dużych opóźnieniach, co jest spowodowane dużą liczbą wymienianych danych (można zastosować pewne rozwiązania, które łagodzą ten problem).

System X Window posiada również inne ograniczenia, które mogą być istotne dla niektórych zastosowań. Pierwszym z nich jest fakt, że współczesne karty graficzne posiadają zaawansowane funkcje graficzne, które nie zawsze można wykorzystać w połączeniu z architekturą omawianego systemu. Kolejny problem mogą stanowić niektóre własności interfejsu użytkownika dostępne dla pulpitów działających lokalnie, a z różnych przyczyn niedostępne lub możliwe do wykorzystania tylko przy spełnieniu pewnych dodatkowych warunków

w systemie X Window (np. współdziałanie aplikacji znajdujących się na jednym pulpicie czy obsługa dźwięku).

1.2. Linux Terminal Server Project[1]

Linux Terminal Server Project (LTSP) został zaprojektowany w celu wykorzystania starych komputerów posiadających niewielkie zasoby sprzętowe jako terminali (klienta). Terminal taki może pracować w jednym z trzech trybów: graficzny interfejs systemu X Window, sesja telnetu lub linia komend. Stacja klienta w momencie startu ładuje do swojej pamięci jądro systemu operacyjnego Linux z jakiegoś nośnika (dyskietka, twardy dysk, CD--ROM, pamięć USB) lub z sieci, wykorzystując protokoły DHCP i TFTP. Następnie, jako korzeń systemu plików podłączana jest za pomocą protokołu NFS odpowiednia gałąź struktury plików, znajdującej się na serwerze. W przypadku trybu terminala graficznego uruchamiana jest sesja systemu X Window przy wykorzystaniu protokołu XDMCP (X Display Manager Control Protocol). System LTSP daje wiele możliwości konfiguracyjnych dodatkowych urządzeń (np. drukarka może być podłączona do stacji klienta). Standardowo aplikacje w tym systemie wykonywane są na serwerze, a jedynie ich interfejs znajduje się po stronie klienta (klient działa wtedy jak klasyczny terminal X Window), przewidziano jednak również możliwość wykonywania aplikacji na komputerze klienta (potrzebny jest wtedy dodatkowo serwer NIS).

1.3. Remote FrameBuffer

W tym protokole stacja, przy której pracuje użytkownik korzystający z urządzeń wejściowych i monitora, zwana jest klientem, zaś komputer, gdzie generowany jest obraz dla monitora – serwerem RFB. Remote Frame Buffer jest protokołem używanym przez VNC (ang. *Virtual Network Computing*). Jest on znacznie prostszy od systemu X Window – działa na niskim poziomie, przesyłając przez sieć skompresowany obraz pulpitu użytkownika [3]. Taki mechanizm pozwala na odłączenie się odległego terminala od serwera, powtórne podłączenie do serwera bez utraty stanu interfejsu użytkownika [6] oraz podłączenie się do tego samego serwera przez wielu klientów równocześnie [7]. Prostota protokołu, zwłaszcza dla komputera klienta sprawia, że jego wymagania sprzętowe są minimalne. Stąd RFB nazywany jest też protokołem „cienkiego klienta" [7]. Zmiany spowodowane działaniami użytkownika nie muszą powodować za każdym razem przesyłania całego obrazu, a jedynie jego fragmentu, który uległ zmianie od ostatniego przesyłu – zmniejsza to ilość danych transmitowanych przez sieć. Dodatkowo zaimplementowano mechanizm adaptacji do

[1] Opracowano na podstawie [5].

przepustowości łącza – wszystkie zmiany zaistniałe na pulpicie są zbierane razem i wysyłane jako całość, przy czym dla wolniejszego łącza przesyły te są robione rzadziej, a dla szybszego częściej [4]. Można powiedzieć, że cała idea protokołu bazuje na pojedynczym graficznym prymitywie: umieść dany piksel w punkcie (x, y). Dane piksela reprezentują jego barwę, przy czym najczęściej stosowany jest format 24-bitowy i 16-bitowy. Najbardziej podstawowym sposobem przesyłu danych obrazu jest surowy format piksel po pikselu. Większość aplikacji protokołu udostępnia też bardziej złożone metody. Przede wszystkim są to grupy pikseli reprezentujące prostokątne wycinki ekranu uzupełnione współrzędnymi położenia, szerokością i wysokością wycinka oraz zastosowaną metodą kompresji (np. algorytmem JPEG) [7].

Założenia protokołu pozwalają na łatwe jego rozszerzanie o przesyłanie informacji o dodatkowych elementach interfejsu użytkownika, np. zmianach kształtu kursora [6]. Protokół jest całkowicie niezależny od systemu operacyjnego, menagera okien i uruchamianych aplikacji [4]. Dla urządzeń wejściowych przewidziano przesyłanie kodów standardowej klawiatury oraz stanu wieloprzyciskowego urządzenia wskazującego (np. myszki) [7]. W przypadku mniej typowych urządzeń wejściowych stosuje się przesyłanie ich danych jako zdarzeń generowanych przez klawiaturę [4]. Protokół ten, dzięki temu, że jest prosty i otwarty, doczekał się licznych implementacji na różnych platformach programowych (wszystkie oparte na systemie X Window, Windows oraz klient w formie apletu, działający na przeglądarkach obsługujących Java) i sprzętowych (istnieje możliwość zdalnego dostępu i sterowania urządzeniami posiadającymi bardzo proste wyświetlacze lub nie mającymi ich w ogóle, np. sprzęt audio-wideo). Najbardziej znanymi implementacjami programowymi są: RealVNC, krdc i krfb oraz Vino. Protokół jest łatwo rozszerzalny o różne dodatkowe elementy (np. algorytmy kodowania obrazu czy różnego rodzaju zabezpieczenia). Działania takie są koordynowane przez Real VNC Ltd., która definiuje kolejne wersje protokołu.

1.4. Remote Desktop Protocol

Protokół ten został opracowany przez firmę Microsoft. Jest on rozszerzeniem Application Sharing Protocol opisanego w dokumencie ITU-T T.128 [8], zwanym też T.SHARE. Jest prostszy od systemu X Window [3], ale bardziej skomplikowany niż RFB [6]. W porównaniu z tym ostatnim działa jednak na wyższym poziomie [3] i dostarcza większy zbiór funkcji, np. przekierowanie drukarki i dźwięku. Protokół ten był i jest rozszerzany przez firmę, która go opracowała. Brak jest ogólnie dostępnych szczegółowych opisów, stąd też trudno nazwać go otwartym. Skutkuje to tym, iż posiada przede wszystkim implementację firmową (usługi terminalowe Windows). Została opracowana również przez niezależnych programistów aplika-

cja klienta dla systemu Linux (rdesktop) na wiedzy zdobytej z przeprowadzonych badań zachowania się firmowej aplikacji w sieci [9].

Sharing Protocol, będący pierwowzorem RDP, definiuje pojęcie pulpitu i okna. Serwer informując klienta o zmianach na ekranie, przesyła informację w postaci skompresowanej bitmapy, przy czym istnieje możliwość kumulacji tych zmian i przesyłania jedynie końcowego efektu dla wolniejszych łączy (podobnie jak w protokole RFB). Opis protokołu przewiduje dwa tryby pracy: dziedziczenia i podstawowy, przy czym tryb podstawowy posiada dodatkowe cechy umożliwiające jego rozszerzanie w przyszłości (co zapewne wykorzystano w kolejnych wersjach protokołu RDP). Różnice pomiędzy trybami polegają głównie na sposobie wymiany danych i negocjacji pomiędzy klientem a serwerem oraz sposobie aktywacji i dezaktywacji klientów. Jako urządzenia wejściowe zostały przyjęte klawiatura i trzyklawiszowe urządzenie wskazujące.

1.5. SSH

SSH (ang. *Secure Shell*) jest zbiorem standardów związanych z szyfrowanym przesyłem danych przez sieć. W szczególności zastępuje on protokół telnet, służący do zdalnej obsługi komputera przez terminal tekstowy. Istnieje również możliwość użycia protokołu SSH do szyfrowania innych protokołów (tzw. tunelowanie). Standardowy system X Window nie szyfruje transmisji, dlatego często jest używany w połączeniu z SSH. Współczesne implementacje dla systemu operacyjnego Linux dają możliwość tunelowania protokołu X11 przez ustawienie odpowiedniej zmiennej w konfiguracji SSH.

1.6. Technologia NX

Technologia NX[2] jest w swojej koncepcji rozwinięciem systemu X Window. Podstawową różnicą w porównaniu do pierwowzoru jest dodanie kompresji danych przy ich przesyle przez sieć. Kompresja ta występuje na trzech poziomach [3]:

- kompresja ruchu sieciowego przez algorytmy kompresji danych, zapis w podręcznej pamięci, stratne i bezstratne algorytmy kompresji obrazów;
- redukcja przesyłów tam i z powrotem;
- adaptacja w czasie rzeczywistym do przepustowości sieci.

Powyższy opis wskazuje, że technologia NX działa najlepiej dla interfejsów, które są oparte na systemie X Window. Aby umożliwić współdziałanie NX z innymi rozwiązaniami informatycznymi, wprowadzono do niego także możliwość enkapsulacji protokołów RDP

[2] Opracowano na podstawie [3].

i RFB. Prócz kompresji danych wprowadzono drugi element, jakim jest szyfrowanie transmisji za pomocą SSH.

2. Porównanie teoretyczne

Porównując protokoły zdalnego dostępu dla systemu operacyjnego Linux pod względem ilości danych przesyłanych przez sieć przy ich zastosowaniu, łatwo zauważyć, że wiele z nich bazuje na systemie X Window. Stąd można przyjąć go jako pewnego rodzaju punkt odniesienia. W podstawowej swojej wersji został on wykorzystany w systemie LTSP. Zastosowanie SSH do tunelowania protokołu X spowoduje zwiększenie ilości danych, które będą przesyłane, gdyż szyfrowanie transmisji nie jest połączone z kompresją danych, a jego zastosowanie będzie wymagało przesyłu dodatkowych informacji związanych z protokołem SSH. Można się zatem spodziewać, że ilość przesyłanych danych w porównaniu z klasycznym systemem X Window wzrośnie, choć nie powinien być to znaczny wzrost. Z kolei technologia NX, zgodnie z intencjami jej autorów, została wzbogacona w porównaniu z systemem X Window o kilka mechanizmów, które mają powodować zmniejszenie ilości danych przesyłanych przez sieć. Stąd można się spodziewać, że algorytmy te zaimplementowane we właściwy sposób przyniosą oczekiwany efekt.

Protokół RFB reprezentuje inne podejście do zagadnienia zdalnego dostępu, stąd bezpośrednie porównanie go z wcześniej zestawionymi systemami jest trudne. Można się spodziewać, że dla różnych systemów i aplikacji ilość przesyłanych danych może być mniejsza raz w przypadku systemu opartego na RFB, innym razem bazującego na X Window. Ten pierwszy przypadek może mieć miejsce, gdy obraz na ekranie będzie bardzo urozmaicony i będzie posiadał wiele różnego rodzaju szczegółów – zakodowanie ich w postaci bitmapy może dać mniej danych niż opisanie go jako elementów systemu X Window. Z kolei wykorzystanie tego ostatniego może dać lepszy efekt, gdy obraz składa się z prymitywów graficznych, które zajmują duży jego obszar.

Protokół RDP w swojej idei jest bardzo zbliżony do protokołu RFB. Różnice dotyczą pewnych założeń i metod zapisywania informacji o zmianach na ekranie. Stąd nie ma prostej odpowiedzi, który z nich będzie wzmagał przesyłania większej ilości danych przez sieć. Dodatkowo sprawę komplikuje fakt, że każdy z tych protokołów może te same dane zakodować na kilka sposobów. Z tego powodu wielkość transmitowanych przez te protokoły danych będzie silnie zależna od ich implementacji zarówno po stronie serwera, jak i po stronie klienta. Tym bardziej trudne jest teoretyczne porównanie protokołu RDP z systemem X Window i jego rozwinięciami.

3. Testy aplikacji

Opisane w tym artykule protokoły posiadają swoje implementacje programowe. Zostały one przebadane pod kątem ilości danych przesyłanych przez sieć dla typowych aplikacji użytkownika przy jednakowych warunkach testowych. Założenia, dobór aplikacji, warunki badań oraz ich wyniki szczegółowo omówiono w [10]. Przeprowadzone testy dla każdego z systemów były następujące (w nawiasach podano oznaczenia testów stosowane w dalszej części artykułu):

- połączenie zdalnego terminala z serwerem (Link),
- otwarcie programu Open Office V2.0.1 (OO),
- otwarcie przeglądarki internetowej FireFox V1.0.7 ze stroną zawierającą tekst o wielkości ok. 8 KB (FFt),
- otwarcie przeglądarki internetowej FireFox V1.0.7 ze stroną zawierającą prócz tekstu dwie grafiki o rozmiarze 15 KB i 45 KB w formacie jpeg (FFg),
- maksymalizacja z paska zadań okna wcześniej zminimalizowanego programu Open Office (OOx),
- maksymalizacja z paska zadań okna ww. przeglądarki ze stroną tekstową (FFtx),
- maksymalizacja z paska zadań okna ww. przeglądarki ze stroną zawierającą grafikę (FFgx).

Spis testowanych połączeń pomiędzy aplikacjami wraz z ich oznaczeniami zawiera tabela 1, zaś uzyskane wyniki tabela 2.

Porównując ilość danych przesyłanych przez sieć przy użyciu systemu LTSP dla środowiska GNOME i KDE, wiedząc równocześnie, że w obu przypadkach używany jest ten sam protokół X, można jednoznacznie stwierdzić, że dla GNOME przesyłanych jest więcej danych – nie mają one jednak istotnego wpływu na wygląd testowanych aplikacji, wynikają więc zapewne z implementacji programowej, która jest daleka od optymalności. Z tego powodu do dalszych porównań przyjęto zasadniczo środowisko KDE.

Zestawienie transmisji SSH i LTSP dla KDE potwierdza przypuszczenia, że szyfrowanie protokołu X spowoduje nieznaczne zwiększenie objętości przesyłanej informacji. Wyniki uzyskane dla technologii NX potwierdzają, że wprowadzone tam mechanizmy kompresji są dosyć skuteczne (szczególnie widoczne jest to dla środowiska GNOME), jednak nie zawsze. Dla testów maksymalizacji okna (OOx, FFgx, FFtx) w środowisku KDE danych jest nieco więcej niż w systemie LTSP. Przyczyną tego może być mała podatność danych na kompresję przesyłanych w tym przypadku, połączona z dodaniem szyfrowania SSH w technologii NX.

Porównanie uzyskanych wyników w środowisku KDE dla LTSP i różnych systemów VNC może prowadzić do stwierdzenia, że protokół RFB przesyła mniej danych niż proto-

kół X, wówczas gdy zmiany na ekranie następują stopniowo (OO, FFg, FFt), gdy zmiany są duże, ale jednorazowe (maksymalizacja okna), to taka sytuacja nie ma już miejsca. Wynika to zapewne z faktu, że protokół X przesyła informację o wszystkich zmianach na ekranie, zaś RFB w przypadku zmian następujących sekwencyjnie po sobie i zaimplementowaniu mechanizmu adaptacji do łącza, może pominąć te zmiany, które są nadpisywane przez kolejne zdarzenia. W systemie Vino można łatwo zauważyć, jak istotne znaczenie ma implementacja teoretycznie opracowanego protokołu – system ten działa w środowisku GNOME i opierając się na tym samym protokole (RFB) przesyła od kilkanastu do kilkadziesięciu razy więcej danych przez sieć niż system RealVNC uruchomiony w takich samych warunkach.

Tabela 1

Spis przeprowadzonych testów

Serwer		Klient		Oznaczenie	Uwagi
Win	Usługi terminal.	Win	rdesktop	rdsk_win	ustawienie: modem 56 Kb/s
Win	Usługi terminal.	Win	rdesktop	rdsk_win_lan	ustawienie: sieć LAN
Win	Usługi terminal.	Linux	rdesktop	rdp	Protokół RDP
Win	Usługi terminal.	Linux	rdesktop	rdp5	Protokół RDP V5
Win	RealVNC	Win	RealVNC	vnc_win	
Win	RealVNC	Linux	RealVNC	vnc_l2w	
Linux	RealVNC	Win	RealVNC	vnc_w2kde	Desktop KDE
Linux	RealVNC	Win	RealVNC	vnc_w2gnome	Desktop GNOME
Linux	RealVNC	Linux	RealVNC	vnc_kde	Desktop KDE
Linux	RealVNC	Linux	RealVNC	vnc_gnome	Desktop GNOME
Linux	RealVNC	Linux	krdc	krdc	Desktop KDE
Linux	Krfb	Linux	RealVNC	krfb	Desktop KDE
Linux	Vino	Linux	RealVNC	vino	Desktop GNOME
Linux	FreeNX	Linux	FreeNX	nx_kde	Desktop KDE
Linux	FreeNX	Linux	FreeNX	nx_gnome	Desktop GNOME
Linux	LTSP	Linux	LTSP	ltsp_kde	Desktop KDE
Linux	LTSP	Linux	LTSP	ltsp_gnome	Desktop GNOME
Linux	Sshd	Linux	ssh	ssh	

Porównanie otrzymanych wyników dla protokołu RDP i różnych wersji VNC pokazuje, że dla tego pierwszego jest przesyłanych mniej danych. Przyczyn tego faktu może być wiele, o czym już wcześniej była mowa. Potwierdzeniem wspomnianych trudności związanych z kilkoma możliwościami zakodowania tych samych danych są rezultaty uzyskane dla protokołu RDP – serwer w kontakcie z wersją firmową klienta (rdsk_win i rdsk_win_lan) transmituje znacznie mniej danych niż dla klienta alternatywnego. Ma to zapewne związek z zaimplementowanymi, a nie opublikowanymi możliwościami protokołu RDP, które potrafi wykorzystać jedynie firmowy klient. Zauważalna jest też mała ilość danych przesyłanych

w protokole RDP przy wykonywaniu operacji maksymalizacji okna. Jest to zapewne skutek wprowadzenia do definicji protokołu pojęcia okna, co umożliwia bardziej zwięzłe zakodowanie operacji na takim obiekcie w porównaniu z protokołem RFB.

Tabela 2

Ilość przesłanych danych w KB przy poszczególnych testach

System	OO	FFg	FFt	OOx	FFgx	FFtx	Link
rdsk_win	27	33	12	8	10	2	50
rdsk_win_lan	28	12	3	4	2	2	50
Rdp	200	348	71	113	6	10	64
rdp5	195	345	69	114	8	9	64
vnc_win	146	761	182	84	567	127	40
vnc_l2w	545	421	140	524	395	127	14
vnc_w2kde	343	635	316	324	523	393	21
vnc_w2gnome	367	593	288	193	559	285	64
vnc_kde	350	625	311	413	823	350	22
vnc_gnome	434	684	320	265	590	278	87
Krdc	361	646	318	305	765	427	1672
Krfb	286	468	203	113	463	144	94
Vino	19383	14102	14932	3345	7585	4999	101
Nx_kde	416	398	319	363	350	313	540
Nx_gnome	833	711	775	373	460	318	675
ltsp_kde	610	1606	704	402	265	262	–
ltsp_gnome	6344	2704	2474	701	729	834	–
ssh	829	1647	1378	×	×	×	7

4. Podsumowanie

Z przeprowadzonej analizy teoretycznej protokołów można wysunąć wniosek, że jednym z kierunków rozwoju protokołów zdalnego dostępu jest wzbogacanie ich o kolejne metody redukcji ilości przesyłanych danych przez sieć oraz algorytmy pozwalające na adaptację natężenia strumienia danych do przepustowości łącza. Stąd też nowsze rozwiązania dają teoretycznie większe możliwości w tym zakresie. Przeprowadzone testy protokołów na aplikacjach przedstawionych wcześniej w większości przypadków potwierdzają tę tezę. Wyniki badań pokazują także, że nawet najlepszy protokół niewłaściwie zaimplementowany, może powodować przesył o wiele większej ilości danych niż teoretycznie dużo gorszy protokół, ale prawidłowo zakodowany.

LITERATURA

1. X.Org Foundation. http://www.x.org.
2. X Window System. http://en.wikipedia.org/wiki/X_Window_System.
3. Pinzari G. F.: Introduction to NX technology. http://nomachine.org/documents/pdf/intr-technology.pdf, 2003.
4. Richardson T., Stafford-Fraser Q., Wood R. W., Hopper A.: Virtual Network Computing, IEEE Internet Computing, Vol. 2, No. 1, 1998.
5. Linux Terminal Server Project. http://www.ltsp.org.
6. Vino. The Remote Desktop Project. http://www.gnome.org/~markmc/remotedesktop.html.
7. Richardson T.: The RFB Protocol. http://www.realvnc.com/docs/rfbprot.pdf, 2005.
8. ITU-T Recommendation, T.128 – Application Sharing. Geneva, 17-27 March 1997.
9. Rdesktop: A Remote Desktop Protocol Client, http://www.rdesktop.org.
10. Stera P.: Zdalny dostęp – przegląd wybranych rozwiązań. Nowe technologie sieci komputerowych, t. 2, s. 251÷259, WKŁ, Warszawa 2006.

Recenzent: Dr inż. Maciej Bargielski

Wpłynęło do Redakcji 15 listopada 2006 r.

Abstract

The paper presents remote access protocols. It focus on volume data transmission. First part describes chosen protocols: X Window system, Linux Terminal Server Project (LTSP), Remote Frame Buffer (RFB), Remote Desktop Protocol (RDP), SSH and NX technology. Second part includes analysis of volume data transmission generates by each of protocols. Next, results of applications test of this protocols are presented (18 tests described in tab. 1; results presented in tab. 2). Then theoretic arguments are compared with application tests. At the end are conclusion – tests confirm theoretic arguments. Another conclusion is that there are bad applications of good protocols.

Adres

Piotr STERA: Politechnika Śląska, Instytut Informatyki, ul. Akademicka 16, 44-101 Gliwice, Polska, piotr.stera@polsl.pl .