

UNIVERZA V LJUBLJANI
FAKULTETA ZA RAČUNALNIŠTVO IN INFORMATIKO

Dean Črnigoj

Izdelava odjemalca NFS za Windows

DIPLOMSKO DELO

VISOKOŠOLSKI STROKOVNI ŠTUDIJSKI PROGRAM PRVE STOPNJE
RAČUNALNIŠTVO IN INFORMATIKA

Mentorica: doc. dr. Mojca Ciglarič

Ljubljana, 2013

To delo je ponujeno pod licenco Creative Commons Priznanje avtorstva - Deljenje pod enakimi pogoji 2.5 Slovenija (ali novejšo različico). To pomeni, da se tako besedilo, slike, grafi in druge sestavine dela kot tudi rezultati diplomskega dela lahko prosto distribuirajo, reproducirajo, uporabljajo, priobčujejo javnosti in predelujejo, pod pogojem, da se jasno in vidno navede avtorja in naslov tega dela in da se v primeru spremembe, preoblikovanja ali uporabe tega dela v svojem delu, lahko distribuira predelava le pod licenco, ki je enaka tej. Podrobnosti licence so dostopne na spletni strani creativecommons.si ali na Inštitutu za intelektualno lastnino, Streliška 1, 1000 Ljubljana.



Izvorna koda diplomskega dela, njeni rezultati in v ta namen razvita programska oprema je ponujena pod licenco GNU General Public License, različica 3 (ali novejša). To pomeni, da se lahko prosto distribuira in/ali predeluje pod njenimi pogoji. Podrobnosti licence so dostopne na spletni strani <http://www.gnu.org/licenses/>.



Št. naloge: 00338/2012

Datum: 04.09.2012

Univerza v Ljubljani, Fakulteta za računalništvo in informatiko izdaja naslednjo nalogo:

Kandidat: **DEAN ČRNIGOJ**

Naslov: **IZDELAVA ODJEMALCA NFS ZA WINDOWS**
NFS CLIENT IMPLEMENTATION FOR WINDOWS

Vrsta naloge: Diplomsko delo visokošolskega strokovnega študija prve stopnje

Tematika naloge:

Cilj dela je izdelava novega odjemalca NFS za uporabo na operacijskem sistemu Windows. Najprej preučite obstoječe izvedbe odjemalcev NFS in primerjajte njihove prednosti in pomanjkljivosti. Zasnujte lastni načrt odjemalca, ki bo odprtokoden, podpira naj najnovejše različice protokola NFS, deluje na čim več različicah Windows in omogoča pregled strežnika NFS v samostojnem brskalniku. Izberite in predstavite primerna orodja in tehnologije za implementacijo, izbiro utemeljite. Opišite tudi specifikacijo in način uporabe protokola NFS. Implementirajte lastnega odjemalca, ga preizkusite in kritično ovrednotite njegove prednosti in slabosti.

Mentor:


doc. dr. Mojca Ciglarič



Dekan:


prof. dr. Nikolaj Zimic

IZJAVA O AVTORSTVU

diplomskega dela

Spodaj podpisani Dean Črnigoj,

z vpisno številko **63070265**,

sem avtor diplomskega dela z naslovom:

Izdelava odjemalca NFS za Windows

S svojim podpisom zagotavljam, da:

- sem diplomsko delo izdelal samostojno pod mentorstvom

doc. dr. Mojca Ciglarič

- so elektronska oblika diplomskega dela, naslov (slov., angl.), povzetek (slov., angl.) ter ključne besede (slov., angl.) identični s tiskano obliko diplomskega dela
- soglašam z javno objavo elektronske oblike diplomskega dela v zbirki »Dela FRI«.

V Ljubljani, dne 19. marca 2013

Podpis avtorja:

Zahvala

Zahvaljujem se svojim staršem za izkazano podporo pri študiju. Še posebej se zahvaljujem mentorici doc. dr. Mojci Ciglarič in Andreju Krevl za mentorstvo in nasvete pri izdelavi diplomske naloge.

Seznam uporabljenih kratic

FTP – File Transfer Protocol

NFS – Network File System

XDR – External Data Representation

ONC – Open Network Computing

RPC – Remote Procedure Call

NLM – Network Lock Manager

RFC – Request for Comments

VM – Virtual Machine

OSI – Open Systems Interconnection

RFC – Request for Comments

IETF – Internet Engineering Task Force

Kazalo vsebine

POVZETEK.....	1
ABSTRACT	2
1 UVOD	3
2 SORODNO DELO IN PRIMERJAVA	5
2.1 WINDOWS SERVICES FOR UNIX	5
2.2 OMNI NFS CLIENT	7
2.3 OPENTEXT NFS CLIENT	8
2.4 PRONFS CLIENT	9
2.5 PRIMERJAVA	10
3 PREDSTAVITEV UPORABLJENIH ORODIJ.....	11
3.1 RAZVOJNO OKOLJE MICROSOFT VISUAL STUDIO 2010	11
3.2 REMOTE TEA.NET.....	13
3.3 WIRESHARK.....	14
3.4 ORACLE VM VIRTUALBOX	15
4 NFS	17
4.1 KAJ JE NFS ?	17
4.1.1 RPC.....	18
4.1.2 XDR.....	19
4.1.3 MOUNT	19
4.1.4 NLM.....	20
4.1.5 Portmap	21
4.2 ZGODOVINA NFS.....	22
4.3 NFS 2	23
4.4 NFS 3	25
4.5 WEBNFS	26
4.6 NFS 4	27
4.7 NFS 4.1	29
4.7.1 pNFS.....	30
4.8 NFS 4.2	32
5 APLIKACIJA NFS CLIENT	35
5.1 OSNOVNE ZAHTEVE IN CILJI APLIKACIJE	35
5.2 REALIZACIJA OSNOVNIH ZAHTEV	35
5.3 RAZVOJ APLIKACIJE.....	36
5.3.1 Priprava testnega okolja.....	36
5.3.2 Priprava programskega okolja	37
5.3.3 Izdelava grafičnega vmesnika.....	37
5.3.4 Izdelava NFS knjižnice.....	39
5.3.5 Izdelava programskega dela aplikacije	41
5.3.6 Testiranje aplikacije.....	41

5.4 PROBLEMI TEKOM IZDELAVE	42
5.5 PREDSTAVITEV APLIKACIJE.....	44
5.6 OBJAVA IZVORNE KODE NA SPLETU	48
6 SKLEPNE UGOTOVITVE	49
LITERATURA.....	51

Kazalo slik

Slika 1: Omogočanje Services for NFS v Microsoft Windows 7 Ultimate.....	5
Slika 2: Omni NFS client z oknom za dodajanje novega preslikanega pogona.....	7
Slika 3: Rezultat programa NFS Trace in prikaz optimalne nastavitve	8
Slika 4: Prikaz programa za nastavitve ter programa za preslikavo v ProNFS client.....	9
Slika 5: Izgled Microsoft Visual Studio v načinu pregledovanja ter programiranja kode	12
Slika 6: Izgled Visual Studio v načinu vizualnega pregledovanja	13
Slika 7: Uporaba orodja jrpcgen.....	14
Slika 8: Pregled shranjene datoteke z aplikacijo Wireshark.....	15
Slika 9: Oracle VM VirtualBox s operacijskim sistemom Ubuntu 10.10	16
Slika 10: Prikaz arhitekture odjemalca-strežnika s protokolom NFS	17
Slika 11: Sklad plasti ISO/OSI modela NFS protokola.....	18
Slika 12: Prikaz vseh registriranih RPC strežnikov z uporabo rpcinfo ukaza.....	22
Slika 13: Razvoj NFS protokola skozi leta ter pripadajoče RFC oznake.....	23
Slika 14: Prikaz arhitekture odjemalca-strežnika s pNFS strežnikom	31
Slika 15: Prikaz programa Norton Commander	35
Slika 16: Prikaz grafičnega vmesnika v zgodnjih fazah izdelave odjemalca NFS.....	38
Slika 17: Primer generatorja naključnih TCP prostih vrat odjemalca.....	43
Slika 18: Omogočanje odjemalca NFS 4.1 v jedru Linux.....	44
Slika 19: Prikaz okna za dodajanje novega strežnika.....	45
Slika 20: Prikaz glavnega okna aplikacije.....	45
Slika 21: Prikaz okna z nastavitvami.....	46
Slika 22: Prikaz okna za kopiranje datotek	47
Slika 23: Prikaz okna za prepisovanje obstoječih datotek.....	47
Slika 24: Prikaz okna za pomoč	48
Slika 25: Prikaz branja vnaprej v NFS protokolu.....	49

Kazalo diagramov

Diagram 1: Diagram poteka povezave na NFS 2 strežnik.....	25
Diagram 2: Diagram poteka povezave na NFS 3 strežnik.....	26
Diagram 3: Diagram poteka povezave s odjemalcem WebNFS	27
Diagram 4: Diagram poteka povezave na NFS 4 strežnik.....	28
Diagram 5: Diagram poteka povezave na NFS 4.1 strežnik.....	30

Povzetek

Glavni cilj diplomskega dela je izdelava aplikacije za operacijski sistem Windows, katera bo služila kot odjemalec NFS protokola. V uvodu je opredeljen problem odjemalca NFS za okolje Windows in razlogi za izdelavo takšnega odjemalca. Opisana so sorodna dela ter njihove pomanjkljivosti in prednosti uporabe. Prav tako so podrobno opisane vse uporabljene tehnologije in orodja, ki so bila uporabljena za dosego cilja. Podrobno je opisan protokol NFS in vse njene različice, kot tudi podporni protokoli ki jih NFS potrebuje. Delo opisuje razvoj aplikacije in probleme, kateri so se pojavili tekom razvijanja. Izdelana aplikacija je namenjena množični uporabi ter nadomesti potrebo po uporabi rešitev predstavljenih v sorodnih delih. Naštete so tudi določene pomanjkljivosti aplikacije ter možnosti za nadaljnji razvoj.

Ključne besede: NFS, Windows, odjemalec, aplikacija, RPC, XDR, NLM

Abstract

The main objective of the thesis is to produce application for Windows, which will serve as an NFS client. The introduction defines the problem of NFS client for Windows, and the reasons for making such a client. Related works are described and their strengths and weaknesses of usage. It also details all of the technology and tools that have been used to achieve the objective. It describes in details the NFS protocol and all of its versions, as well as the protocols that support NFS needs. The thesis describes the development of application and problems which have emerged during development. Designed application is intended for widespread use and replace the need to use the solution presented in the related work. It also describes some weaknesses of the application and opportunities for further development.

Keywords: NFS, Windows, client, application, RPC, XDR, NLM

1 Uvod

Potreba po oddaljenemu dostopu do datotek sega v sam začetek povezovanja računalnikov v omrežje. Protokol FTP omogoča prenos datotek, sam dostop do oddaljene datoteke pa je bil pred prenosom le te na odjemalčev datotečni sistem, onemogočen. Z razvojem protokola NFS se je omogočil dostop do oddaljenih datotek v omrežjih brez potrebe, da bi se le te prenesle na računalnik. Če se osredotočimo na odjemalec NFS, je ta zelo dobro podprt v operacijskih sistemih UNIX, podpora na operacijskem sistemu Windows pa je slabša, ker ta za skupno uporabo datotek uporablja lasten protokol CIFS. Uporabnik Windows operacijskega sistema ima z dostopom do NFS strežnika kar nekaj težav. Našteli bomo nekaj dejstev zakaj je temu tako:

- Obstoječi odjemalec, katerega je razvil Microsoft, deluje, vendar ne na vseh operacijskih sistemih Windows.
- Skoraj vsak prodan računalnik že vsebuje instaliran operacijski sistem Windows in nima zmožnosti uporabe privzetega Microsoft odjemalca NFS.
- Plačljivi odjemalci so na voljo, vendar po njih ponavadi posegajo predvsem podjetja.
- Navaden uporabnik si poišče alternativo v Samba strežniku namesto obstoječega NFS.
- Problem nastane, če uporabnik hoče dostopati do strežnika NFS, kateri ni njegov ter nima možnosti namestitve Samba strežnika.

Omenjena dejstva so povod k potrebi, da se razvije aplikacija za operacijski sistem Windows, ki bi nudila podporo za NFS protokol.

Za dosego cilja je bilo potrebno razumevanje NFS protokola ter znanje izdelovanja Windows aplikacij. NFS protokol je podrobno opisan v RFC dokumentih, za izdelavo aplikacije pa smo uporabili orodje Microsoft Visual Studio 2010, katero je opisano v poglavju 3.1. Celotna aplikacija je bila izdelana samostojno, v veliko pomoč pa nam je prišlo orodje jrcpgen ter Wireshark. Končni produkt je odjemalec NFS, kateri ima podporo za najnovejši protokol NFS ter zmožnosti pregledovanja, kopiranja, brisanja in ustvarjanja datotek na NFS strežniku.

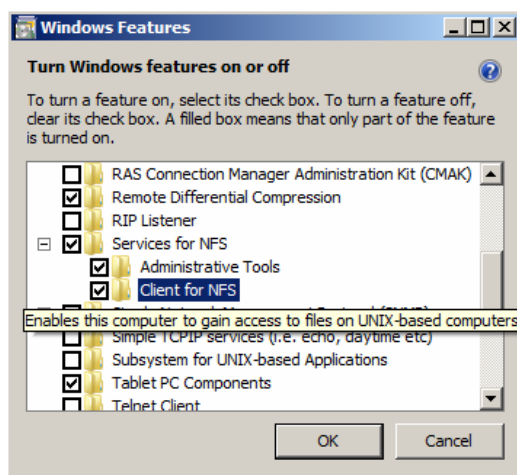
2 Sorodno delo in primerjava

Aplikacije z odjemalcem za NFS protokol in podporo operacijskemu sistemu Windows obstajajo že dolgo časa. V tem poglavju bomo poskušali predstaviti sorodne aplikacije in njihove pomanjkljivosti oziroma prednosti v primerjavi z diplomskim delom.

2.1 Windows Services for UNIX

Windows services for UNIX je dodatek za operacijske sisteme Windows, ki ga je razvil Microsoft in je na voljo kot instalacijska datoteka na internetu. Podpira operacijske sisteme: Windows 2000, Windows Server 2003 ter Windows XP Professional.

Dodatek za novejšje operacijske sisteme Windows se imenuje Services for NFS in je opcijsko na voljo v operacijskih sistemih: Windows Server 2003 R2, Windows Server 2003 R2 x64, Windows Server 2008, Windows Vista Ultimate, Windows Vista Enterprise, Windows 7 Ultimate in Windows 7 Enterprise. V obeh primerih dodatek doda enake funkcionalnosti odjemalca NFS. Slika 1 prikazuje kako omogočimo Services for NFS v operacijskem sistemu Windows 7 Ultimate.



Slika 1: Omogočanje Services for NFS v Microsoft Windows 7 Ultimate

Z namestitvijo Windows services for UNIX ali omogočanjem Services for NFS namestimo odjemalec z podporo NFS 2 in NFS 3. Odjemalec preslika skupen imenik NFS strežnika v prost omrežni pogon. To lahko naredimo vizualno prek mojega računalnika ter dodaj omrežni pogon ali pa ročno z uporabo programa mount.

```
mount 192.168.0.13:/export/shared/ Z:
```

Ta ukaz nam preslika skupni imenik `/export/shared/` na NFS strežniku z omrežnim naslovom 192.168.0.13 v omrežni pogon Z.

Pred tem pa moramo še določiti anonimen uporabniški identifikator ter skupinski identifikator, sicer nam NFS strežnik povezavo zavrne, saj se Windows predstavi z identifikatorjem uporabnika kateri je prijavljen trenutno v sistem.

Uporabniški identifikator in Skupinski identifikator nastavimo tako da v register dodamo na mestu:

```
HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\ClientForNFS\CurrentVersion\Default\
```

Dodamo dve novi spremenljivki tipa `DWORD` 32 in sicer `AnonymousUid` ter `AnonymousGid`. Ko smo zapisali spremembe v register, napišemo v konzolo (`cmd`) dva ukaza za ponoven zagon odjemalca NFS:

```
nfsadmin client stop
nfsadmin client start
```

S to proceduro omogočimo povezovanje na NFS strežnik z uporabo poljubnega uporabniškega identifikatorja in skupinskega identifikatorja sistema.

Pregled vseh skupnih imenikov na NFS strežniku je omogočeno z uporabo programa `showmount`.

Prednosti uporabe Windows services for UNIX so naslednje:

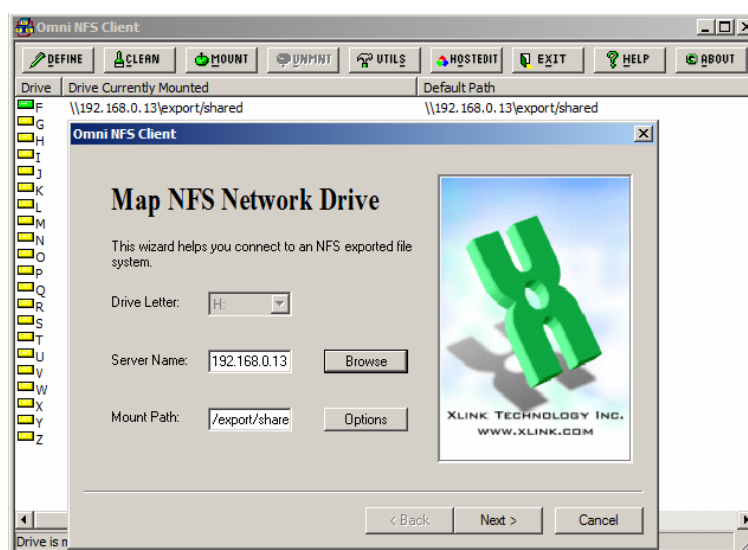
- Brezplačen.
- Odjemalec ima lahko namestitvev, še posebno v novejših operacijskih sistemih.
- Samodejna detekcija protokola TCP in UDP ter različice 2 ali 3.

Slabosti pa so naslednje:

- Ni podpore NFS 4 in NFS 4.1.
- Ne podpira vseh operacijskih sistemov Windows. Primer je Windows 7 Home Premium.
- Omogoča pregledovanje samo preko omrežnega pogona.

2.2 Omni NFS client

Omni NFS client je razvilo podjetje Xlink. Omni NFS client podpira protokol NFS 2 in NFS 3. Namestitev je mogoča na operacijske sisteme od Windows 95 naprej. Omogoča prikaz prostih mrežnih pogonov ter preslikovanje skupnih imenikov na NFS strežniku na le te. Omogoča tudi samodejno zaznavanje NFS strežnikov v lokalnem omrežju. Celotna uporaba programa poteka preko grafičnega vmesnika in je zelo enostavna za uporabo. Slika 2 prikazuje grafični vmesnik odjemalca, kot tudi vodič za preslikovanje NFS strežnika v omrežni pogon.



Slika 2: Omni NFS client z oknom za dodajanje novega preslikanega pogona

Prednosti uporabe Omni NFS client so naslednje:

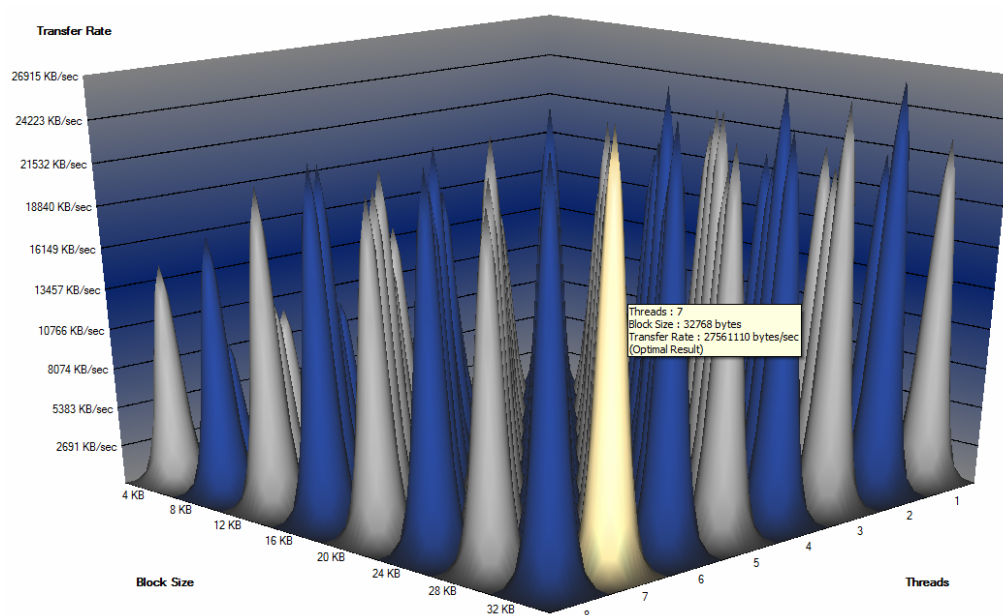
- Enostaven za uporabo.
- Podpira skoraj vse Microsoft Windows operacijske sisteme.
- Samodejna zaznava NFS strežnikov v lokalnem omrežju.

Slabosti:

- Ni podpore NFS 4 in NFS 4.1.
- Plačljiv (119\$, marec 2013).
- Omogoča pregledovanje samo preko omrežnega pogona.

2.3 OpenText NFS Client

OpenText NFS Client je izdelalo podjetje OpenText. Podpira protokole NFS 2, NFS 3 in NFS 4, prav tako pa podpira tudi NLM protokol ter veliko varnostnih protokolov. Odjemalec je preprost in omogoča preslikavo skupnega imenika v omrežni pogon. Poleg samega odjemalca dobimo v paketu tudi ogromno orodij za konfiguracijo ter diagnostiko NFS protokola. Še posebej zanimivo je orodje NFS Trace, katero testira NFS strežnik z prebiranjem ter pisanjem v datoteko z različnimi nastavitvami ter potem izbere najboljšo kombinacijo in le to po želji nastavi v Odjemalec NFS.



Slika 3: Rezultat programa NFS Trace in prikaz optimalne nastavitve

Prednosti uporabe OpenText NFS Client so naslednje:

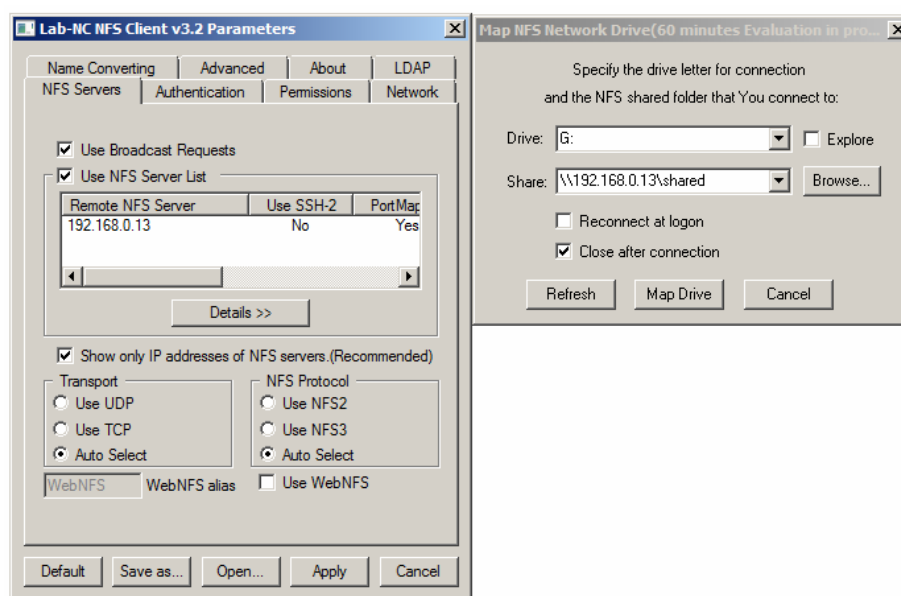
- Podpora NFS 4.
- Samodejna zaznava NFS strežnikov v lokalnem omrežju.
- Ogromno dodatnih programov.

Slabosti:

- Ni podpore NFS 4.1.
- Plačljiv (245\$, marec 2013).
- Omogoča pregledovanje samo preko omrežnega pogona.

2.4 ProNFS client

ProNFS client je odjemalec NFS, ki ga je izdelalo podjetje Labtam. Odjemalec podpira NFS 2 ter NFS 3 protokol. Prav tako podpira NLM 3 in NLM 4 protokol za zaklep datotek. Podpora Windows operacijskih sistemov je podprta od različice Windows 95 naprej. Sam odjemalec ima dva programa in sicer program za namestitve ter program za preslikavo skupnih imenikov NFS strežnikov v omrežne pogone. Ponuja tudi veliko dodatnih programov za diagnostiko, kot je na primer NFS Probe, kateri poda RPC informacije o danem računalniku v omrežju.



Slika 4: Prikaz programa za nastavitve ter programa za preslikavo v ProNFS client

Prednosti uporabe ProNFS client so naslednje:

- Veliko nastavitev.
- Podpira skoraj vse Microsoft Windows operacijske sisteme.

Slabosti:

- Ni podpore NFS 4 in NFS 4.1.
- Plačljiv (40\$, marec 2013).
- Omogoča pregledovanje samo preko omrežnega pogona.

2.5 Primerjava

Kot vidimo je vsak izmed odjemalcev imel nekaj pomanjkljivosti. Noben izmed štirih testiranih odjemalcev ni imel podpore protokolu NFS 4.1. Vsi razen Windows services for UNIX so bili plačljivi, le ta pa ni imel podpore za vse operacijske sisteme Windows. Vsi odjemalci so podpirali preslikavo skupnega imenika v omrežni pogon, noben pa ni podpiral samostojnega brskalnika. Želimo torej Odjemalec NFS, ki bo imel naslednje lastnosti:

- Brezplačen in odprtokoden.
- Podpora zadnjim različicam protokola NFS.
- Enostaven za uporabo.
- Deluje na čim več operacijskih sistemih Windows.
- Omogoča pregled NFS strežnika v samostojnem brskalniku.

3 Predstavitev uporabljenih orodij

Pri izbiri glavnega razvojnega okolja smo se odločili za uporabo okolja Microsoft Visual Studio 2010, saj je tudi cilj naloge izdelati odjemalec za uporabnike v okolju Windows. Ker smo kot programski jezik izbrali C#, smo potrebovali orodje Remote Tea.Net, katero je XDR opis protokola pretvorilo v C# datoteke. Za zajem omrežnih paketov smo uporabljali orodje Wireshark. Ker smo potrebovali tudi NFS strežnik, smo le tega namestili na Linux operacijski sistem, katerega smo virtualizirali s programom Oracle VM Virtual Box.

3.1 Razvojno okolje Microsoft Visual Studio 2010

Visual Studio[1] je integrirano razvojno okolje, ki ga je razvil Microsoft. Prva različica je bila razvita leta 1995 in še ni vključevala podpore vsem programskim jezikom, ki jih je Microsoft podpiral. Takšna podpora je bila dodana v naslednjih različicah. Za razvoj aplikacije smo uporabili različico 2010.

Različica 2010 omogoča razvoj:

- Windows Forms aplikacij.
- Konzolnih aplikacij.
- Aplikacij z grafičnim vmesnikom.
- Spletnih strani.
- Spletnih aplikacij.
- Spletnih storitev.

Razvoj je podprt na naslednjih platformah:

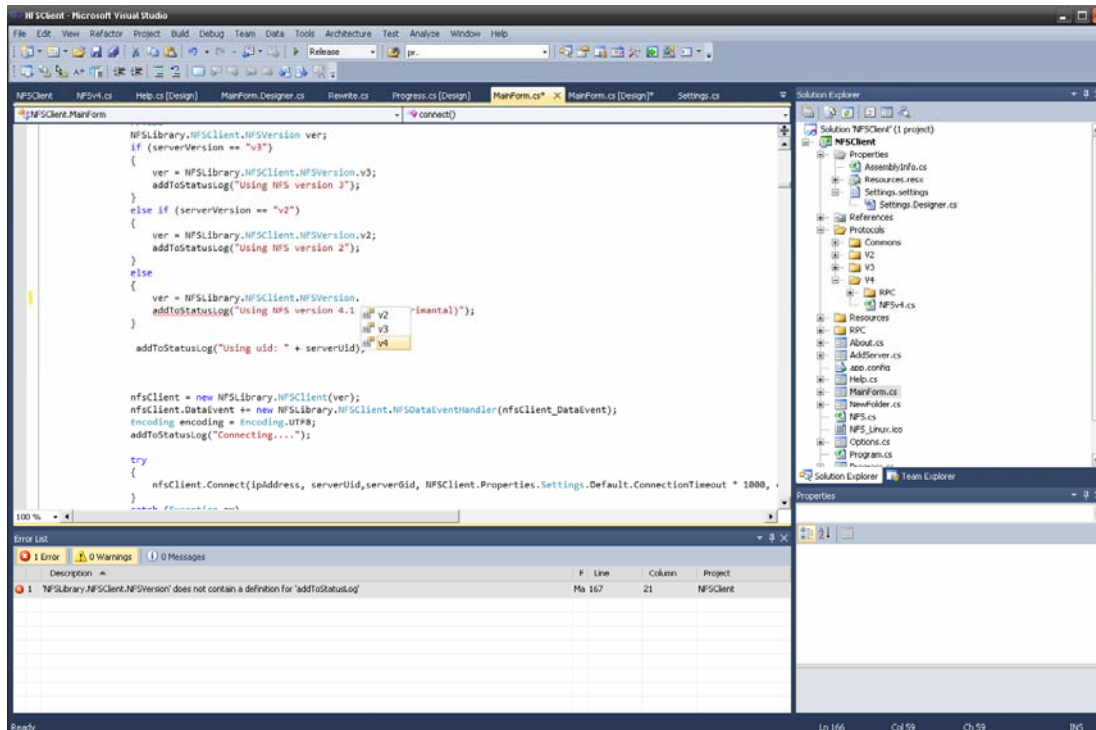
- Microsoft Windows
- Windows CE
- Windows Mobile
- .NET Framework
- .NET Compact Framework
- Microsoft Silverlight

Omogoča programiranje v različnih programskih jezikih. Ti so:

- C/C++
- VB.NET
- C#
- F#
- preko servisov M,Python in Ruby
- XML/XSLT
- HTML/XHTML
- JavaScript
- CSS

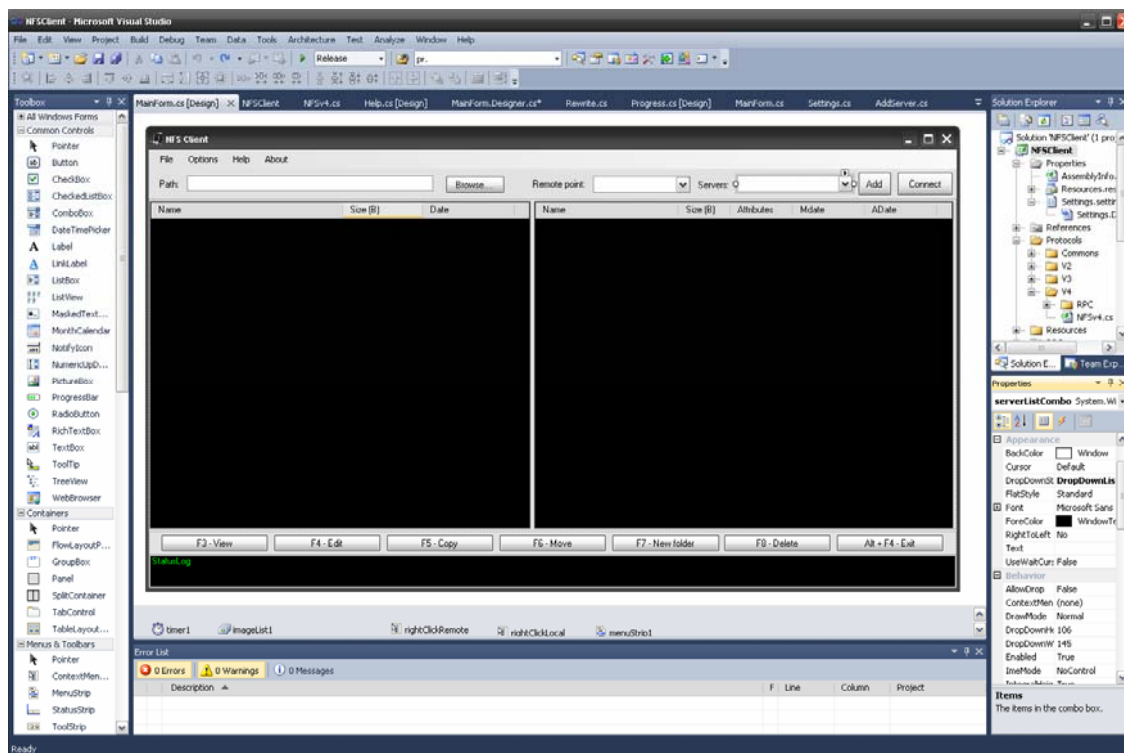
Naša aplikacija je nastala s pomočjo Windows Forms, ki je namenjen za uporabo na operacijskem sistemu Microsoft Windows, kot programski jezik pa smo izbrali C#. Aplikacija je bila razvita s pomočjo ogrodja Microsoft .NET Framework različice 4, ki je tudi najnovejše podprto s strani Visual Studio.

Slika 5 prikazuje pogled Visual Studio v načinu pregledovanja ter programiranja kode. Kot vidimo nam je pri programiranju v veliko pomoč IntelliSense, ki nam po pregledu nabora funkcij in metod, med pisanjem kode omogoča samo dokončevanje.



Slika 5: Izgled Microsoft Visual Studio v načinu pregledovanja ter programiranja kode

Okolje Visual Studio pa omogoča še pogled vizualnega načrtovanja. V sliki 6 bomo prikazali kako poteka načrtovanje Windows Forms aplikacije v tem pogledu. Pogled nam omogoča dodajanje ter spreminjanje lastnosti različnih kontrol, katere lahko uporabimo na Windows Forms aplikaciji. S pomočjo panelov smo lahko aplikaciji določili, da se s spreminjanjem velikosti osnovnega okna, okni ki vsebujeta seznam datotek, razširita ter obdržita razmerje velikosti.



Slika 6: Izgled Visual Studio v načinu vizualnega pregledovanja

3.2 Remote Tea.Net

Remote Tea.Net[2] je orodje, ki omogoča popolno implementacijo protokola ONC/RPC v jeziku C#. Izvira iz Remote Tea[23] projekta, katero omogoča enako implementacijo, vendar v programskem jeziku Java. Oba orodja sta brezplačna ter omogočata nadaljnjo razvijanje. Za izdelavo diplomskega dela smo uporabljali tudi jrpcgen, kateri je del orodja Remote Tea.Net, ki nam je iz opisa XDR le tega pretvoril v C# datoteke. Jrpcgen orodje je nadomestek orodja rpcgen, katerega je razvil Sun Microsystems in je omogočal pretvorbo v jeziku C. Kot zanimivost naj omenimo, da je Remote Tea dodal črko J pred to orodje, Remote Tea.Net pa je

to poimenovanje pustil, saj bi bilo logično, da le to preimenujejo v na primer Crpcgen ali kaj podobnega.

Slika 7 opisuje možnosti klica, ter uporabo orodja Jrpcgen in sicer za protokole NFS različice 2,3 in 4.1.

```

C:\WINDOWS\system32\cmd.exe
where options include:
-c <classname> specify class name of client proxy stub
-d <dir> specify directory where to place generated source code files
-p <package> specify package name for generated source code files
-s <classname> specify class name of server proxy stub
-ser tag generated XDR classes as serializable
-bean generate accessors for usage as bean, implies -ser
-noclamp do not clamp version number in client method stubs
-withcallinfo supply call information to server method stubs
-initstrings initialize all strings to be empty instead of null
-nobackup do not make backups of old source code files
-noclient do not create client proxy stub
-noserver do not create server proxy stub
-parseonly parse x-file only but do not create source code files
-verbose enable verbose output about what jrpcgen is doing
-version print jrpcgen version and exit
-debug enables printing of diagnostic messages
-? -help print this help message and exit
-- end options

C:\remotetea\trunk\jrpcgen\bin\Release>jrpcgen.exe nfsv2.x
C:\remotetea\trunk\jrpcgen\bin\Release>jrpcgen.exe NFSv3Protocol
C:\remotetea\trunk\jrpcgen\bin\Release>jrpcgen.exe NFSv4Protocol.x
C:\remotetea\trunk\jrpcgen\bin\Release>

```

Slika 7: Uporaba orodja jrpcgen

3.3 Wireshark

Wireshark[3] je brezplačna aplikacija za analizo omrežnega prometa. Uporablja se za reševanje omrežnih težav, analizo ter programski in omrežni razvoj programov. Wireshark je aplikacija, ki deluje na različnih operacijskih sistemih. Podpira Linux, Mac OS X, BSD, Solaris ter Microsoft Windows.

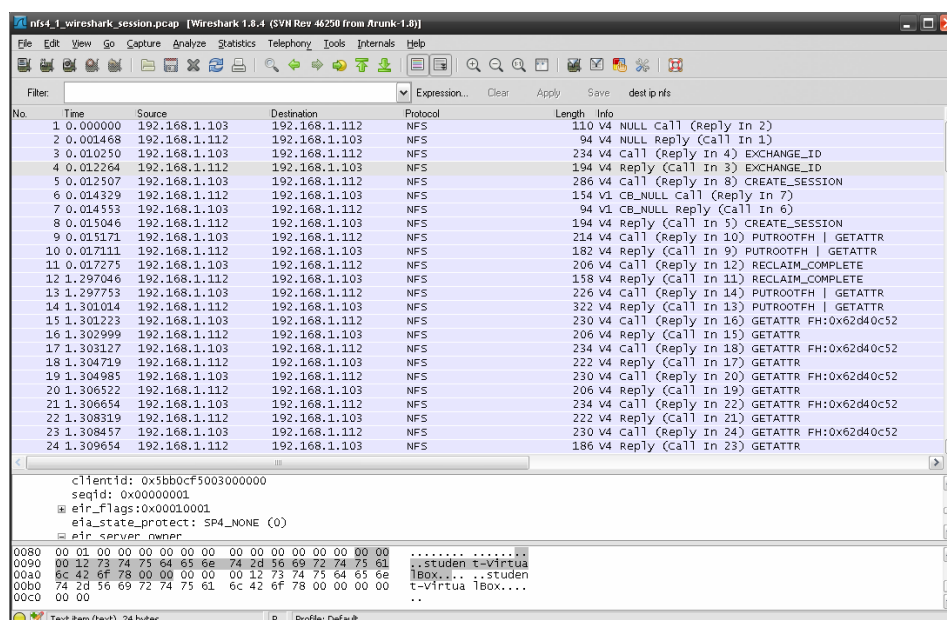
Aplikacija Wireshark omogoča:

- Podatki se lahko zajemajo 'iz žice' v živo iz omrežne povezave ali pa prebrani s datoteke ki je bila že vnaprej zajeta.
- Podatki v živo so lahko zajeti iz različnih omrežij kot naprimer Ethernet, IEEE 802.11, PPP in loopback.
- Zajeta omrežni paketi se lahko pregledajo preko naprednega uporabniškega vmesnika ali iz konzolnega programa TShark.
- Prikazani podatki se lahko filtrirajo z uporabo različnih, že napisanih filtrov.
- Omogoča uporabo dodatkov za zajem novih protokolov.

- Omogoča zajem ter zaznavo paketov VOIP. Če so ti dekodirani, so lahko tudi predvajani.

Wireshark uporablja za shranjevanje zajetih podatkov format libpcap, kateri je podprt z strani tudi mnogih drugih aplikacij.

Slika 8 prikazuje Wireshark z odprto datoteko z že zajeti podatki. Podatki so bili zajeti v Ubuntu operacijskemu sistemu in prikazujejo povezavo med NFS strežnikom ter odjemalcem različice 4.1. Z datoteko smo lahko na sistemu Windows, kjer smo imeli tudi nameščeno omenjeno aplikacijo, še dodatno analizirali zajete pakete ter nato razvili podporo protokolu.



Slika 8: Pregled shranjene datoteke z aplikacijo Wireshark

3.4 Oracle VM VirtualBox

Oracle VM VirtualBox[4] je orodje za virtualizacijo x86 operacijskih sistemov. Podpira operacijske sisteme Linux, Mac OS X, Microsoft Windows, Solaris ter Open Solaris. Omogoča virtualizacijo operacijskih sistemov Microsoft Windows, Linux, BSD, OS/2 in Solaris. Razvit je bil v podjetju Innotek GmbH, sedaj pa ga aktivno razvija podjetje Oracle.

Glavne značilnosti orodja so:

- Podpora 64 bitni operacijskim sistemom, če je na voljo strojna podpora za virtualizacijo (Intel VT-x ali AMD-V).
- Omogoča posnetke (Shranjevanje programskega stanja operacijskega sistema da ob naslednji uporabi ni potreben ponoven zagon).
- Skupno odlagališče (Uporaba odlagališča je skupna na gostujočem kot TUDI gostiteljevemu operacijskemu sistemu).
- Deljeni imeniki.
- Podpora VMDK ter VHD diskovnim datotekam (VMDK je diskovna datoteka VMware orodja, VHD pa orodja Virtual PC).
- Direkten dostop do diska gostiteljevega računalnika.
- 3D grafična podpora (OpenGL ter DirectX podpora).

Slika 9 prikazuje orodje Oracle VM VirtualBox, na katerem je virtualiziran Linux operacijski sistem Ubuntu 10.10, katerega smo uporabljali za razvoj aplikacije.

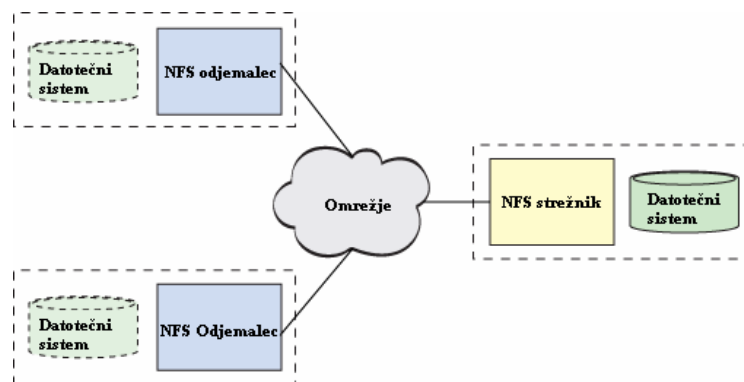


Slika 9: Oracle VM VirtualBox s operacijskim sistemom Ubuntu 10.10

4 NFS

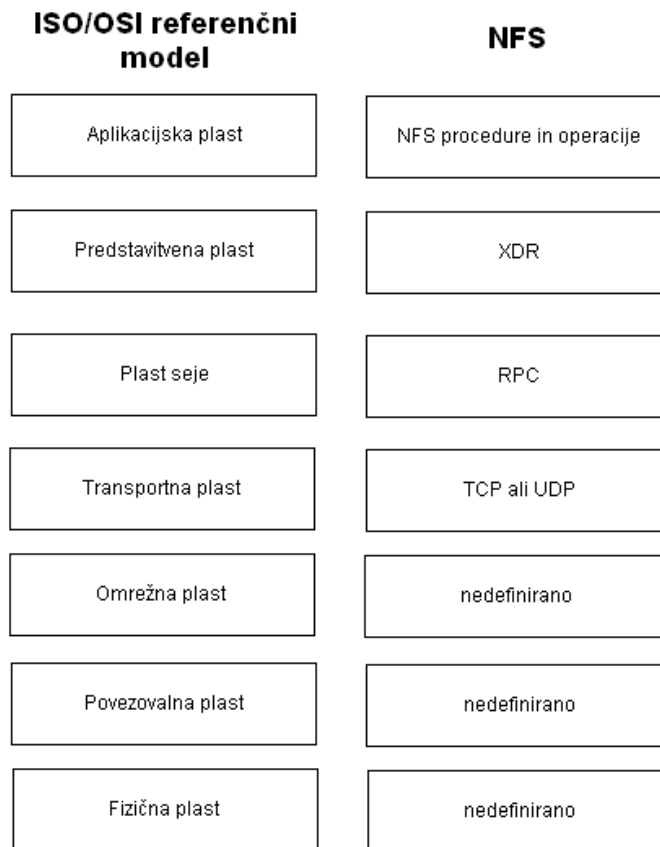
4.1 Kaj je NFS ?

NFS[5,6] je protokol porazdeljenega datotečnega sistema, katerega je razvilo podjetje Sun Microsystems pribl. leta 1980. Omogoča uporabniku oddaljeni dostop do datotek preko omrežja s pomočjo odjemalca ter strežnika. Za razliko od FTP protokola, ki je bil razvit pred protokolom NFS, omogoča NFS protokol dostop do datoteke, brez da bi le to prej prenesel na sam datotečni sistem odjemalca. Tako lahko odjemalec pregleduje datoteke tudi brez uporabe lastnega datotečnega sistema.



Slika 10: Prikaz arhitekture odjemalec-strežnik s protokolom NFS

NFS protokol je grajen s pomočjo ONC/RPC, katerega je Sun Microsystems izdal skupaj s NFS protokolom. Slika 11 Prikazuje sklad plasti NFS protokola v primerjavi z referenčnim modelom ISO/OSI.



Slika 11: Sklad plasti ISO/OSI modela NFS protokola

Protokol posluša na TCP in UDP vratih 2049, odjemalec pa mora zahteve pošiljati z vrat 665 do 1023 sicer lahko v nasprotnem primeru NFS strežnik proceduro zavrne. NFS protokol je večinoma uporabljen v operacijskih ki izhajajo iz UNIX-a, vendar lahko strežnik ali odjemalec namestimo tudi v okolju Windows. Če izvzamemo prvo različico, je NFS protokol prosto dostopen ter dobro dokumentiran, tako da je izdelava odjemalca ali strežnika po standardih NFS omogočena vsakomur. Kot vidimo je NFS protokol grajen s pomočjo veliko drugih protokolov in standardov. Le te bomo opisali v naslednjih poglavjih.

4.1.1 RPC

RPC[7] je protokol katerega najdemo v plasti seje OSI modela. Omogoča odjemalcu oddaljen klic procedure, katera se izvede na strežniku. Podobno kot klic procedure v katerekoli programskem jeziku omogoča podajanje parametrov ter vrača rezultate procedure. Odjemalec protokola RPC točno ve katere parametri morajo biti podani kot tudi kakšne vrste rezultat bo prinesel klic procedure. Z uporabo protokola RPC je programiranje modela odjemalec-

strežnik lažje, saj odpade načrtovanje sporočil med samim odjemalcem ter strežnikom. Vsak klic odjemalca sproži na strežniku točno določeno proceduro, tako da ostane samo še načrtovanje procedur ter klicev odjemalca. RPC omogoča, da strežnik podpira več različnih programov kot različic naenkrat. Tako lahko preko RPC protokola imamo strežnik, ki podpira starejšo različico, kot tudi novejšo različico programa. RPC protokol je razvil Bruce Jay Nelson, podjetje ki je protokol prvo uporabilo, pa je bilo podjetje Xerox Corporation in sicer leta 1981. RPC je postal poznan prav s pomočjo implementacije v protokolu NFS. Podrobno delovanje je opisano v dokumentu RFC 1050[8], RPC 2 protokol pa v dokumentu RFC 5531[9].

4.1.2 XDR

XDR[10] je standard za predstavitev podatkov in njeno uporabo v omrežjih. Omogoča da se lahko podatki prenesejo v različne računalniške sisteme. Pretvorba iz lokalnih spremenljivk ter struktur v XDR se imenuje kodiranje, pretvorba iz XDR v lokalne spremenljivke ter strukture pa se imenuje dekodiranje. XDR je implementiran kot programska knjižnica in omogoča uporabo v različnih operacijskih sistemih ter neodvisnost od transportnega protokola. XDR uporablja bloke velikosti 32 bitov. Tudi če je tip spremenljivke velik samo 1 bit, zasede po kodiranju 32 bitov. Uporablja pravilo debelega konca in za večje podatkovne strukture kot so sezname ali niz črk dodaja bloke tako, da se kodiranje vedno izide. Podpira veliko podatkovnih tipov ter za decimalna števila uporablja IEEE 754 format. XDR je bil razvit v podjetju Sun Microsystems pribl. leta 1980. Leta 1987 pa je bil javnosti prvič predstavljen v dokumentu RFC 1014[11]. XDR je postal IETF standard leta 1995, ko je bil dopolnjen v dokumentu RFC 1832[12]. Najnovejši dokument, ki opisuje XDR standard je RFC 4506[13], a ta ne prinaša nobenih tehničnih sprememb v primerjavi s prejšnjim dokumentom.

4.1.3 MOUNT

MOUNT[5] je podporni protokol NFS 2 in NFS 3. Uporablja se izključno v navezi z NFS protokolom. Prav tako kot NFS, se protokol nahaja v aplikacijski plasti OSI modela, vendar za pravilno delovanje potrebuje XDR v predstavitveni plasti ter RPC v plasti seje. Za delovanje uporablja UDP ali TCP kot transportni protokol, vrata namenjena MOUNT protokolu pa niso določena, tako da je potrebno pred vsakim prvim klicem MOUNT uporabiti servis Portmap, ki pove na katerih vratih posluša strežnik MOUNT. Obstajajo tri različice protokola MOUNT, vendar med njimi ni bistvene razlike, razen v rezultatu procedure MNT katera vrne v MOUNT 3 64 bitni identifikator imenika, v MOUNT 1 pa 32 bitni id imenika.

NFS 2 uporablja MOUNT 1 in MOUNT 2, NFS 3 pa MOUNT 3. RPC identifikator MOUNT protokola je 100005, sam protokol pa vsebuje šest procedur MOUNT:

- NULL - ne naredi ničesar, samo za poizvedovanje dosegljivosti MOUNT strežnika.
- MNT – vrne id (angl. file handle) dovoljenega imenika, kateri je podan kot argument procedure (strežnik si zapolni da je odjemalec zavzel ta imenik).
- DUMP – vrne seznam vseh povezanih odjemalcev in njihov omrežni naslov (ukaz `showmount` na Unix operacijskem sistemu vrne podoben rezultat).
- UMNT – sprosti shranjenega odjemalca.
- UMNTALL – sprosti vse povezane odjemalce.
- EXPORT – vrne seznam dovoljenih imenikov na NFS strežniku.

4.1.4 NLM

NLM[5] protokol je podporni protokol NFS 2 in NFS 3. Uporablja se izključno v navezi z NFS protokolom, vendar njegova uporaba ni nujna. Omogoča zaklep datotek in s tem preprečuje branje oziroma pisanje v datoteko drugim aplikacijam, razen NFS strežniku. Odjemalec se odloči ali bo NLM protokol uporabljal ali ne, vendar je uporaba priporočljiva. Prav tako kot NFS, se protokol nahaja v aplikacijski plasti OSI modela, vendar za pravilno delovanje potrebuje XDR v predstavitveni plasti ter RPC v plasti seje. Za delovanje uporablja UDP ali TCP kot transportni protokol, vrata namenjena NLM protokolu pa niso določena, tako da je potrebno pred vsakim prvim klicem NLM uporabiti servis Portmap, ki pove na katerih vratih posluša strežnik NLM. RPC identifikator NLM protokola je 100021, protokol pa ima štiri različice. NLM 1 in NLM 3 protokola se uporabljata v NFS 2, NLM 4 pa v NFS 3 protokolu. Razlika med NLM 1 in NLM 2 je predvsem v dodatnih procedurah za odjemalce v ne UNIX okolju. Razlika med NLM 3 in NLM 4 pa je v 64 bitnem identifikatorju datoteke. Glavne procedure, katere NLM protokol uporablja so naslednje:

- NULL – ne naredi ničesar, samo za poizvedovanje dosegljivosti NLM strežnika.
- TEST – poizvede ali bo zaklep uspel ali ne (procedura se v praksi ne uporablja saj je lahko med uspešnem odgovoru o zaklepu in dejanskem zaklepu datoteka že zaklenjena).
- LOCK – zakleni datoteko.
- UNLOCK ali CANCEL – sprosti zaklenjeno datoteko.
- GRANTED – procedura katera je izvedena iz strežnika in sporoči odjemalcu da je datoteka ki jo je prej zahteval in je bila zavrnjena sedaj na voljo.

4.1.5 Portmap

Portmap[5, 21] ali Rpcbind je RPC servis, kateri poda informacije o drugih RPC strežnikih, kateri so na voljo. Uporablja UDP in TCP transportni protokol in posluša na vratih 111. Vsak RPC strežnik se mora ob svojem začetku registrirati v Portmap ter podati informacije o različici programa, uporabljenem protokolu TCP ali UDP ter vratih na katerih bo poslušal. Tako lahko odjemalec povpraša na katerih vratih posluša strežnik NFS, MOUNT, NLM ali drugi RPC strežnik. Servis je opisan v RFC 1833 dokumentu. RPC procedure ki jih uporablja so naslednje:

- NULL – ne naredi ničesar, samo za poizvedovanje dosegljivosti Portmap servisa.
- SET – registrira strežnik, podani atributi morajo biti id programa, različica programa, vrsta transportnega protokola in vrata na katerih bo strežnik poslušal.
- UNSET – odstrani registriran strežnik iz seznama.
- GETPORT – s to proceduro odjemalec povpraša na katerih vratih posluša dani RPC strežnik.
- DUMP – vrne seznam vseh registriranih strežnikov vključno z vsemi kombinacijami transportnih protokolov.
- CALLIT – omogoča odjemalcu da preko Portmap pokliče določeno proceduro na RPC strežniku, brez da bi vedel na katerih vratih strežnik posluša. Rezultat je samo ali se je procedura pravilno izvršila.

Seznam vseh registriranih strežnikov lahko pridobimo tudi lokalno s klicem `rpcinfo -p`. Slika 12 prikazuje seznam vseh registriranih RPC strežnikov na našem testnem okolju.

```

student@student-VirtualBox:/export/shared$ rpcinfo -p
  program vers proto  port
100000    2   tcp    111  portmapper
100000    2   udp    111  portmapper
100021    1   udp   55119 nlockmgr
100021    3   udp   55119 nlockmgr
100021    4   udp   55119 nlockmgr
100021    1   tcp   35639 nlockmgr
100021    3   tcp   35639 nlockmgr
100021    4   tcp   35639 nlockmgr
100003    2   tcp   2049  nfs
100003    3   tcp   2049  nfs
100003    4   tcp   2049  nfs
100227    2   tcp   2049  nfs
100227    3   tcp   2049  nfs
100003    2   udp   2049  nfs
100003    3   udp   2049  nfs
100003    4   udp   2049  nfs
100227    2   udp   2049  nfs
100227    3   udp   2049  nfs
100005    1   udp  46308 mountd
100005    1   tcp  59918 mountd
100005    2   udp  46308 mountd
100005    2   tcp  59918 mountd
100005    3   udp  46308 mountd
100005    3   tcp  59918 mountd

```

Slika 12: Prikaz vseh registriranih RPC strežnikov z uporabo rpcinfo ukaza.

4.2 Zgodovina NFS

NFS protokol ima že več kot 30 letno zgodovino delovanja in nadgrajevanja. Prva različica je bila na voljo samo znotraj podjetja Sun Microsystems, zato lahko samo ugibamo, kdaj točno so bili začetki razvoja protokola. S prihodom NFS 2 leta 1984, se je protokol začel zelo hitro razširjati, saj je bila dokumentacija za razvoj le tega prosto dostopna v dokumentu RFC 1094[14].

Ker je imel NFS 2 številne pomanjkljivosti, kot na primer podpora datotekam večjim od 2 GB, se je začel razvoj NFS 3. Protokol NFS 3 je bil tako dokončan leta 1995 in je bil v skladu s navodili, izboljšava NFS 2. Njegove ključne prednosti so bile:

- Podpora datotekam večjim od 2 GB.
- Podpora asinhronnega pisanja na strežnik.
- Dodatna REaddirPLUS operacija.
- Podpora transportnemu protokolu TCP.

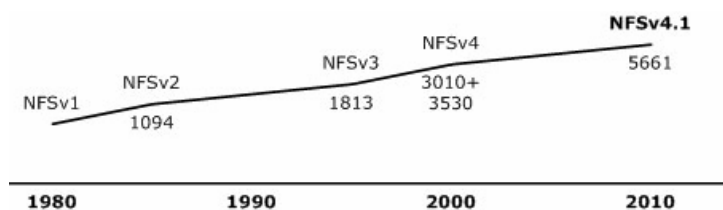
Zaradi specifikacij RPC je lahko strežnik podpiral tudi več različic, tako da je NFS 3 strežnik podpiral tudi NFS 2 in s tem ohranil podporo starejšim odjemalcem.

Leta 1995 se je protokol uporabljal večinoma v lokalnih omrežjih in s hitrim razvojem interneta so se pojavile številne pomanjkljivosti. Zaradi tega je bila leta 1996 izdana WebNFS[19] različica katera je poenostavila uporabo na internetu.

Nato je Sun Microsystems predal razvoj protokola IETF organizaciji. Ta je skupaj s Sun Microsystems leta 2000 razvila NFS 4 protokol. V dokumentu RFC 3010[15], kateri je bil kasneje dopolnjen v dokumentu RFC 3530[16], je opisana četrta različica NFS protokola. Protokol je bil na novo napisan, imel je samo dve proceduri in sicer COMPOUND ter NULL, in je podpiral tri različne varnostne sisteme. Druge značilnosti so še:

- Zahteva po uporabi transportnega protokola TCP.
- Podpora zaklepa datotek v samem protokolu.
- Integriran MOUNT protokol v protokolu NFS.
- Podpora povratnih informacij odjemalcu s procedurama CB_COMPOUND in CB_NULL.
- Podpora UTF-8.
- Možnost dodajanja operacij v naslednjih pod različicah (angl. minor versions).

Zadnji izmed različic NFS protokola, pa je različica 4.1, katera prinaša podporo paralelnega dostopa, tako imenovani pNFS. Razvit je bil leta 2010 in je opisan v RFC 5661[17] dokumentu.



Slika 13: Razvoj NFS protokola skozi leta ter pripadajoče RFC oznake

4.3 NFS 2

NFS 2[5, 14] je bil prvi NFS protokol predstavljen javnosti. Za transportni protokol uporablja UDP, za seznam vseh dostopnih imenikov na NFS strežniku pa uporablja protokol MOUNT. Za zaklepanje datotek uporablja NLM 1 in NLM 3 protokol. NFS 2 strežnik ne vsebuje

nikakršnih informacij o povezanih odjemalcih in ne posodablja stanja odjemalca (angl. Stateless). Vsak odjemalec ki je povezan na strežnik se vsakič znova predstavi. Z uporabo procedure WRITE ali READ lahko prenesemo največ 8192 bajtov podatkov, datoteke pa lahko prebiramo do največ 2GB velikosti. Najpomembnejše procedure, ki jih ta protokol podpira so:

- NULL – ne naredi ničesar, samo za poizvedovanje dosegljivosti strežnika.
- GETATR – pridobi attribute datoteke glede na id objekta (angl. file handle).
- SETATR – nastavi attribute glede na id objekta (angl. file handle).
- LOOKUP – preveri ali datoteka v podani mapi obstaja in vrne pravice le te.
- READ – prebere datoteko podano na id objekta (angl. file handle).
- CREATE – ustvari prazno datoteko in vrne id objekta (angl. file handle).
- WRITE – piše v datoteko katera je bila ustvarjena prej.
- REMOVE – izbriše kazalec na datoteko podano v imeniku (če je to edini kazalec le to tudi izbriše).
- RENAME – preimenuje datoteko.
- MKDIR – ustvari imenik, poda se tudi attribute.
- RMDIR – izbriše imenik, a le če je ta prazen.
- READDIR – poda vse datoteke v imeniku podanem s id objekt(angl. File handle).

Klic vsake procedure vrne status, če je ta NFS_OK(0), je klic procedure uspel in je rezultat pravilen, v nasprotnem primeru pa vrne identifikator napake.

Diagram 1 prikazuje, kakšen vrstni red procedur je potreben, da se uspešno povežemo na NFS 2 strežnik ter pridobimo seznam datotek v prvem dovoljenem imeniku. Procedure so iz MOUNT in NFS 2 protokola.

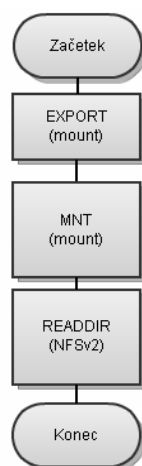


Diagram 1: Diagram poteka povezave na NFS 2 strežnik

4.4 NFS 3

NFS 3[5, 18] je nadgradnja protokola NFS 2. Za transportni protokol uporablja UDP ali TCP, za seznam vseh dostopnih imenikov na NFS strežniku pa uporablja protokol MOUNT. Z uporabo procedure WRITE ali READ lahko prenesemo število podatkov določeno s proceduro FSINFO, datoteke pa lahko prebiramo poljubnih velikosti. Spremembe procedur v primerjavi z prejšnjo različico so naslednje:

- LOOKUP – vsebuje še attribute imenika.
- READ – vrne tudi status ali je prebran odsek tudi konec datoteke.
- READDIRPLUS – v nasprotju s READDIR, le ta poda dodatno informacijo o id datoteke in njene attribute.
- FSINFO – procedura poda informacije kot so največji prenos podatkov z READ ali WRITE ter želeni prenos in ali strežnik podpira simbolične povezave ter trde povezave.
- FSSTAT – procedura poda informacije o datotečnem sistemu NFS strežnika kot so velikost ter koliko prostora je porabljenega.
- COMMIT – procedura v navezi z asinhronim WRITE potrjuje sinhronizacijo.

MOUNT protokol je bil rahlo spremenjen zaradi 64 bitnega identifikatorja datoteke. Prav tako je zaradi tega protokola bil razvit nov NLM 4 protokol.

Diagram 2 prikazuje kakšen vrstni red procedur je potreben, da se uspešno povežemo na NFS 2 strežnik ter pridobimo seznam datotek v prvem dovoljenem imeniku. Procedure so iz MOUNT in NFS 3 protokola.

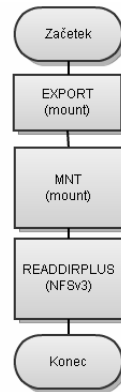


Diagram 2: Diagram poteka povezave na NFS 3 strežnik

4.5 WebNFS

Sama različica protokola WebNFS[19] ne prinaša veliko sprememb. Uporaba je predvidena na strežniku NFS 3, predvideva pa uporabo vrat 2049, uporabo transportnega protokola TCP in uporabo skupnega identifikatorja. Torej odpade uporaba MOUNT protokola ter klicanje Portmap servisa. Skupni identifikator za različico NFS 2 vsebuje 32 ničel, za različico NFS 3 pa prazen identifikator z vsebino nič bitov. Podporo strežniku WebNFS se preveri z klicanjem tega skupnega identifikatorja, kateri je vnaprej dogovorjen. Če je odgovor strežnika, da datoteka ne obstaja, WebNFS podpora ni podprta ali pa skupni identifikator ni določen. Sprememba je tudi v proceduri LOOKUP, saj v standardu WebNFS le ta podpira več nivojsko preverjanje. Kot primer navedimo star način poizvedbe datoteke a/b/c:

```

LOOKUP  FH=0x0  "a"  --->
                                <---  FH=0x1
LOOKUP  FH=0x1  "b"  --->
                                <---  FH=0x2
LOOKUP  FH=0x2  "c"  --->
                                <---  FH=0x3
  
```

Z uporabo protokola WebNFS pa lahko to preverimo z enim samim klicem:

```

LOOKUP  FH=0x0  "a/b/c" --->
                                <---  FH=0x3
  
```

Uporaba WebNFS je bila predvidena predvsem za branje datotek, ampak je lahko uporabljena tudi za pisanje in spreminjanje le teh. Primarno je bil standard razvit za uporabo v Java applet aplikacij. Sam protokol se je leta 2007 preimenoval v YANFS (angl. Yet Another NFS).

Diagram 3 prikazuje kakšen vrstni red procedur ter operacij je potreben, da se uspešno povežemo s odjemalce WebNFS ter pridobimo seznam datotek v skupnem identifikatorju. Kot vidimo se v nasprotju s NFS 3 ali NFS 2 število klicev zmanjša na ena, vendar moramo predvideti, da mogoče NFS strežnik ne podpira WebNFS ali pa skupni identifikator ni definiran.

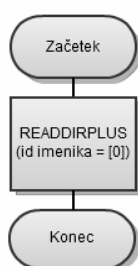


Diagram 3: Diagram poteka povezave s odjemalcem WebNFS

4.6 NFS 4

NFS 4[5, 16] je nadgradnja protokola NFS 3 in NFS 2. Uporablja RPC 2 protokol, ki je podrobno opisan v dokumentu RFC 5531[9]. Za transportni protokol uporablja UDP ali TCP. Za varnost protokola je poskrbljeno z uporabo GSS-API. Izbiramo lahko med tremi vrstami zaščite, in sicer Kerberos 5, LIPKEY ali SPKM-3. Za zaklepanje datotek ter izpis seznama je poskrbljeno v samem protokolu NFS, tako da ni potrebe po dodatnih protokolih. Protokol ima samo dve proceduri, in sicer NULL in COMPOUND ter povratne procedure CB_NULL in CB_COMPOUND. Če se uporablja povratne procedure, mora odjemalec biti tudi strežnik in poslušati na dogovorjenih vratih za povratne informacije. Namesto tipov procedur, ki jih uporabljajo prejšnje različice NFS protokola, uporablja NFS 4 samo eno proceduro COMPOUND, ki pa ima več operacij katere so podobne proceduram iz prejšnjih različic. Hkrati lahko kličemo več operacij naenkrat, kot primer lahko pridobimo id datoteke ter njene attribute in dostop. Struktura klica COMPOUND procedure je naslednji:

```

+-----+-----+-----+-----+-----+-----+-----+
| tag | minorversion | numops | op + args | op + args | op + args |
+-----+-----+-----+-----+-----+-----+-----+

```

Tag je poljubno poimenovanje klica, ki je večinoma prazno, minorversion pa je podatek o podani pod različici protokola, v našem primeru 0. Sledi število operacij ter njen id in argumenti. Rezultat klica pa je naslednji:

```
+-----+-----+-----+-----+-----+
|last status | tag | numres | status + op + results |
+-----+-----+-----+-----+-----+
```

Če so se vse operacije izvršile brez napake, je rezultat COMPOUND procedure NFS4_OK(0), v nasprotnem primeru pa je rezultat napaka, vendar je status posebej vrnjen za vsako operacijo.

Seznam imenikov, ki so na voljo, se pridobi s klicem operacije PUTROOTFH ter GETFH, kateri vrnejo identifikator korenskega imenika. Tako imamo na voljo pregled do celotnega datotečnega sistema, a le če je ves dovoljen v nastavitvah NFS strežnika. Kot primer naj navedemo uporabnika ki ima dostop do imenika / ter /export/shared, vendar nima dostopa do /export. Tako ne bo mogel pridobiti vsebine imenika /export/shared, saj nima dostopa do vseh pod imenikov. Za razliko bi s protokolom NFS 2 in NFS 3 lahko tako dostopal do imenika /export/shared, a le če bi bil ta zapisan kot skupen imenik in bi identifikator imenika pridobil s pomočjo MOUNT protokola.

Diagram 4 prikazuje kakšen vrstni red procedur ter operacij je potreben, da se uspešno povežemo na NFS 4 strežnik ter pridobimo seznam datotek v korenskem imeniku. Kot vidimo se v nasprotju s NFS 3 ali NFS 2 število klicev zmanjša, saj lahko v enem klicu procedure uporabimo več operacij hkrati.

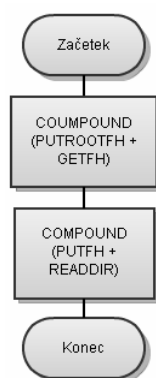


Diagram 4: Diagram poteka povezave na NFS 4 strežnik

4.7 NFS 4.1

NFS 4.1[17] je nadgradnja protokola NFS 4. Vsebuje vse specifikacije protokola četrte različice in le tega dopolnjuje s novimi operacijami ter posodobljenimi iz prejšnje različice. Spremembe s protokolom četrte različice so naslednje:

- Podpora sejam.
- Paralelni dostop do podatkov (pNFS).
- Dodatna operacija RECLAIM_COMPLETE za boljšo strukturo zaklepanja datotek.
- Atributi so lahko podani avtomatsko ob ekskluzivnem kreiranju datotek s zastavico EXCLUSIVE4_1.

Procedura COMPOUND ostaja enaka, le spremenljivka minorversion je nastavljena na 1, kar nakaže strežniku, da se hočemo povezati s protokolom različice štiri in prve pod različice. Zaradi podpore sejam, strežnik zahteva najprej predstavitev odjemalca in s tem določitev enoličnega identifikatorja odjemalca z operacijo EXCHANGE_ID. Nato je potrebno ustvariti sejo z operacijo CREATE_SESSION in v atribut podati identifikator odjemalca. Po ustvarjeni seji mora odjemalec z vsakim klicem katerekoli operacije kot prvo operacijo podati SEQUENCE, v kateri se kot atribut poda številko klicev od ustvarjene seje. Prav tako mora odjemalec v primeru neaktivnosti, vsakih 60 sekund poslati prazno SEQUENCE operacijo, katera potrdi da je odjemalec še vedno aktiven. V nasprotnem primeru bo vsak naslednji klic vrnil napako NFS4ERR_BAD_SESSION in sejo bo potrebno ponovno ustvariti.

Diagram 5 prikazuje kakšen vrstni red procedur ter operacij je potreben, da se uspešno povežemo na NFS strežnik različice 4.1 ter pridobimo seznam datotek v korenskem imeniku. Kot vidimo, se število klicev poveča, saj je v nasprotju s protokolom četrte različice potrebno ustvariti enolični identifikator ter sejo in le to vzdrževati da ne zastara.

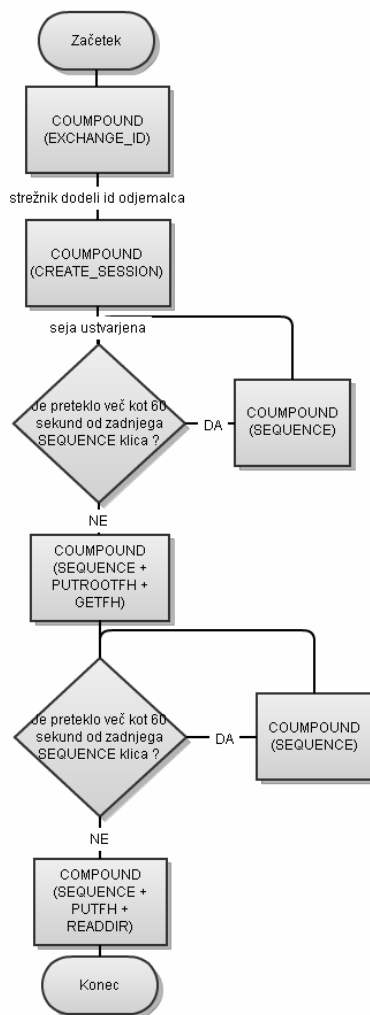
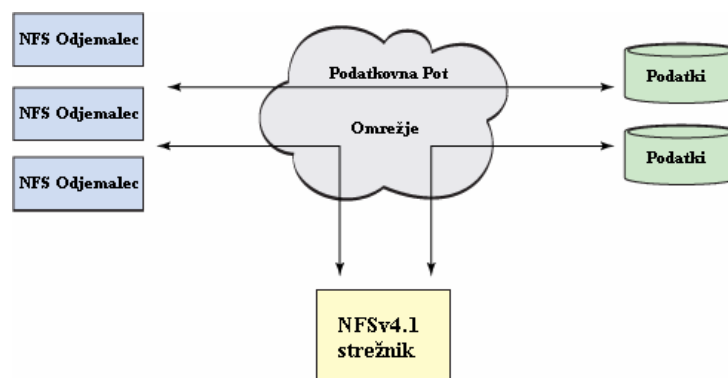


Diagram 5: Diagram poteka povezave na NFS 4.1 strežnik

4.7.1 pNFS

Prejšnje različice NFS so vse predstavljale samo datotečni sistem, ki je bil na voljo na samem strežniku. S prihodom različice 4.1 nam NFS strežnik omogoča vpogled v več podatkovnih virov, kateri niso nujno fizično prisotni na samem NFS strežniku. Predstavi se paralelni dostop do podatkov na NFS strežniku ali krajše pNFS. V tem primeru NFS strežnik posreduje samo meta podatke, kot so imena datotek, attribute ter kje se datoteka nahaja, same datoteke ali bloke datotek pa odjemalec dobi s podatkovnih strežnikov. NFS strežnik obenem še poskrbi za preverjanje pristnosti odjemalcem. NFS strežnik je lahko strežnik meta podatkov kot podatkovni strežnik naenkrat, vendar je to brez dodatnih podatkovnih strežnikov nesmiselno, saj je neuporaba pNFS enostavnejša. Podatkovni strežnik lahko podpira NFS

protokol, ali pa druge podatkovne protokole kot so Internet SCSI (iSCSI) ali FCP (Fibre Channel Protocol for SCSI).



Slika 14: Prikaz arhitekture odjemalec-strežnik s pNFS strežnikom

Podporo pNFS odjemalec zaprosi z zastavicami v operaciji `EXCHANGE_ID`. Če odjemalec noče uporabe pNFS postavi zastavico `EXCHGID4_FLAG_USE_NON_PNFS` na 1. V nasprotnem primeru odjemalec postavi zastavico `EXCHGID4_FLAG_USE_PNFS_MDS`, če hoče biti predstavljen NFS strežnik kateri bo predstavljal meta podatke ali `EXCHGID4_FLAG_USE_PNFS_DS`, če hoče predstavljal samo datoteke.

```
+-----+
| Pričakovani rezultati zastavic v operaciji EXCHANGE_ID |
+-----+
| EXCHGID4_FLAG_USE_PNFS_MDS |
| EXCHGID4_FLAG_USE_PNFS_MDS | EXCHGID4_FLAG_USE_PNFS_DS |
| EXCHGID4_FLAG_USE_PNFS_DS |
| EXCHGID4_FLAG_USE_NON_PNFS |
| EXCHGID4_FLAG_USE_PNFS_DS | EXCHGID4_FLAG_USE_NON_PNFS |
+-----+
```

Tabela prikazuje možne rezultate NFS 4.1 strežnika. Te rezultate strežnik potrdi in odjemalec jih mora spoštovati ter na rezultate le teh odločati o nadaljnji vrsti povezave s strežnikom.

Za podporo pNFS strežnik potrebuje uporabo dodatnih operacij:

- `GETDEVICEINFO` - vrne naslov podatkovnega strežnika glede na njegov id.

- GETDEVICELIST – vrne naslov vseh podatkovnih strežnikov za podan datotečni sistem.
- LAYOUTGET – vrne id podatkovnega strežnika ter sporoči strežniku da se bo datoteka brala ali pisala (datoteko ali blok zavzame).
- LAYOUTCOMMIT – sporoči strežniku z meta podatki da se je v datoteko pisalo podatke.
- LAYOUTRETURN – sporoči strežniku z meta podatki da je končalo s branjem ali pisanjem datoteke (datoteko ali blok sprosti).

Uporabljajo pa se tudi povratne operacije in sicer:

- CB_LAYOUTRECALL – sporoči odjemalcu na sprosti določeno datoteko, vse datoteke določenega datotečnega sistema ali vse datoteke zavzete s strani odjemalca.
- CB_RECALL_ANY – sporoči odjemalcu naj sprosti določeno število zavzetih datotek ali blokov, operacija se ponavlja kliče samo če je strežnik preobremenjen.
- CB_RECALLABLE_OBJ_AVAIL – pove odjemalcu, da ja zadnja datoteka ali blok, ki je bil zavrnjen z operacijo LAYOUTGET sedaj na voljo.
- CB_NOTIFY_DEVICEID – sporoči odjemalcu da se je id podatkovnega strežnika spremenil.

4.8 NFS 4.2

V razvoju je NFS 4.2 protokol. Zadnja različica protokola je opisana v 17 osnutku[20]. Za primerjavo omenimo da je protokol NFS 4.1 dosegel 29 osnutkov, prej kot je bil uradno izdan. NFS 4.2 naj bi prinesel številne novosti kot so:

- Podpora strežniški kopiji (angl. Server-side copy) – v prejšnjih različicah je bilo potrebno podatke prvo prenesti k odjemalcu ter nato prekopirati na drug strežnik. Ta novost bo podpirala neposredno kopiranje med strežniki.
- Odjemalci bojo lahko strežniku svetovali kakšen način dostopa do datotek ali blokov lahko pričakuje. (angl. Application I/O advise) z uporabo IO_ADVISE.
- Podpora datotekam z luknjami (angl. Sparse files) – Uporabno za datoteke VHD in VMDK. Razvoj operacij READ_PLUS in WRITE_PLUS katere bodo lahko luknje prenesli z eno operacijo in samo opisom luknje. Primer: prenesi 8943 bajtov ne lociranega prostora.

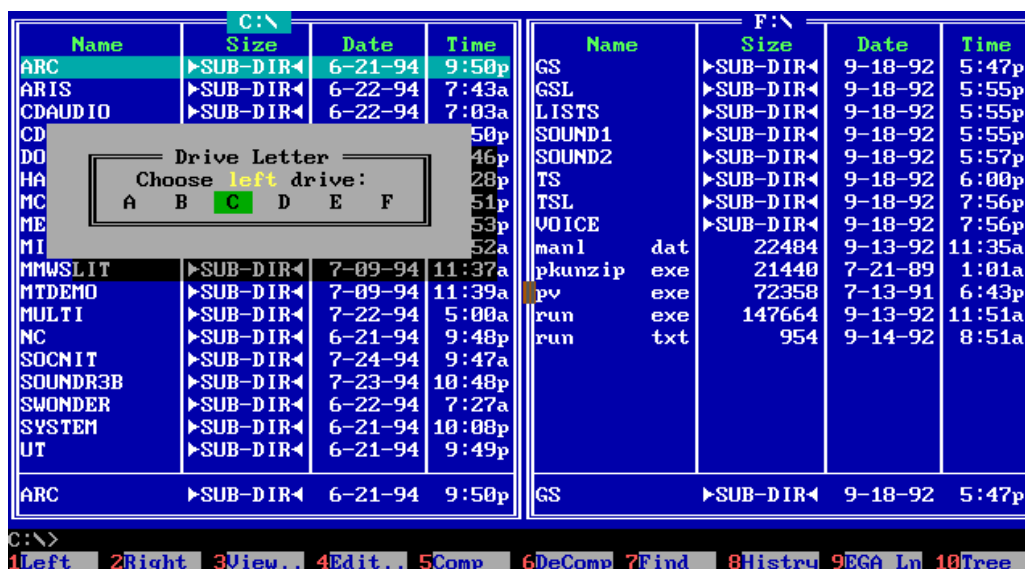
- Predhodna rezervacija prostora – omogoča predhodno rezervacijo prostora, še prej kot se datoteka zapiše na strežnik. Predvsem uporabno za datoteke z luknjami.
- Podpora datotekam katere predstavljajo navidezni pogon – Z operacijami READ_PLUS in WRITE_PLUS bo mogoče te datoteke tudi formatirati ter brati datotečni sistem v datotekah.

To so glavne novosti, katere so bile našteje z osnutkom 17. Do same uradne predstavitve protokola se lahko le te spremenijo, predvsem pa doda še kakšne nove funkcionalnosti.

5 Aplikacija NFS Client

5.1 Osnovne zahteve in cilji aplikacije

Namen izdelave aplikacije je izdelati odjemalec za NFS strežnik, ki bi deloval na Windows operacijskemu sistemu. Hkrati so zahteve tudi, da je aplikacija brezplačna, enostavna za uporabo ter vsebuje podporo za najnovejše protokole NFS. Aplikacija naj bo prenosljiva ter naj omogoča podporo čim več različic operacijskega sistema Windows. Aplikacija naj se zgleduje po Norton Commander programu, katera naj ima v levem oknu prikazan lokalni datotečni sistem, v desnem pa naj le ta prikazuje vsebino NFS strežnika.



Slika 15: Prikaz programa Norton Commander

Dodatno naj aplikacija omogoča shranjevanje že vnaprej shranjenih strežnikov ter avtomatsko povezavo do le teh. Aplikacija naj omogoča dodatne bližnjice ter kombinacije tipk za napredne uporabnike, ter le te vedno obvešča o napredku.

Opcijsko naj aplikacija omogoča hibridni način delovanja in sicer kot brskalnik opisan v osnovnih zahtevah, ali kot omrežni pogon, ki datotečni sistem v NFS strežniku prikaže na operacijskem sistemu Windows.

5.2 Realizacija osnovnih zahtev

Odjemalec je realiziran s pomočjo orodja Visual Studio 2010 in C# programskega jezika ter Windows Forms aplikacije. Podpora Windows operacijskim sistemom je omogočena od različice Windows XP naprej, saj je to zahteva .NET Framework 4.0, s katerim orodje Visual Studio 2010 deluje. Odjemalec podpira protokole NFS 2, NFS 3 in NFS 4.1, kateri je trenutno najnovejši NFS protokol. Odjemalec podpira nastavitve uporabniškega identifikatorja ter skupinskega identifikatorja kateri se povezuje na NFS strežnik. Odjemalec ima dve pod okni, v katerima se prikazujeta lokalni datotečni sistem ter oddaljeni datotečni sistem NFS. Izbira imenika na lokalnem datotečnem sistemu je mogoča z uporabo gumba prebrskaj (angl. browse) ali z izbiro na določeno mapo v seznamu map in datotek. Izbira imenika na oddaljenem datotečnem sistemu je omogočena z izbiro na določeno mapo ali s spremembo skupnega imenika v izvlečnem meniju. Odjemalec je realiziran v angleškem jeziku, saj je njegov namen, da bi se ga uporabljalo po celem svetu. Prav tako je koda odjemalca kot tudi komentarji realizirana v angleškem jeziku saj je odjemalec pripravljen za nadaljnje odprto kodno razvijanje. Uporaba odjemalca je enostavna, na voljo pa je tudi pomoč.

Shranjevanje že uporabljenih strežnikov je realizirano z uporabo nastavitve aplikacije Windows Forms. Aplikacija podpira operacije s pomočjo gumbov na spodnji strani aplikacije, z kombinacijo tipk, s pomočjo izvlečnega menija ali s pomočjo povleči-spusti. Aplikacija ima na spodnji strani seznam zadnjih operacij, tako da je uporabnik vedno obveščen o napredku, za kopiranje datotek pa se odpre še novo okno katero bolj podrobno nakaže napredek.

Odjemalec podpira tudi prikaz skupnih imenikov NFS strežnika v omrežnem pogonu. Ta podpora je omogočena z uporabo Dokan knjižnice. Ko izberemo način prikaza strežnika v omrežnem pogonu, se aplikacija pomanjša v statusno vrstico ter prikaže vsebino omrežnega pogona.

5.3 Razvoj aplikacije

Razvoj aplikacije je potekal v več fazah razvoja. Od priprave okolja, do izdelave grafičnega vmesnika ter NFS knjižnice. Testiranje aplikacije je privedlo do večkratne ponovitve zadnjih treh faz, saj je šele testiranje pokazalo določene napake, zaradi katerih je bilo potem potrebno spreminjanje grafičnega okolja ali NFS knjižnice.

5.3.1 Priprava testnega okolja

Za razvoj aplikacije smo najprej potrebovali testno okolje, saj je bilo protokol potrebno najprej analizirati. Odločili smo se za Linux operacijski sistem, kateri je imel nameščen NFS strežnik kot odjemalec. Izbrali smo distribucijo Ubuntu in sicer različico 10.10. Namestili smo ga v Oracle VM VirtualBox ter namestili NFS strežnik z uporabo naslednjih vrstic kode:

```
sudo apt-get install nfs-kernel-server
sudo mkdir -p /export/share
sudo chmod 777 export/ -R
sudo pico /etc/exports
sudo pico /etc/default/nfs-common
sudo pico /etc/default/nfs-kernel-server
sudo /etc/init.d/nfs-kernel-server restart
```

Tako smo namestili NFS strežnik ter naredili imenik `/export/share` ter v `/etc/exports` dodali `/export/shared *(rw,subtree_check)` vrstico ki omenjeni imenik doda v skupno rabo za vse uporabnike. V `/etc/default/nfs-common` ter `/etc/default/nfs-kernel-server` smo določili da strežnik ter odjemalec ne uporablja nobene avtorizacije. Odjemalca NFS nam ni bilo potrebno namestiti, saj je že ta vključen v jedro Linux operacijskega sistema. Naj še omenimo, da smo za namestitev odjemalca različice 4.1 potrebovali na novo prevesti jedro, saj je primarno odjemalec 4.1 onemogočen, ker je še v eksperimentalnem delovanju. Nato smo namestili program Wireshark, ki smo ga potrebovali za zajem paketkov.

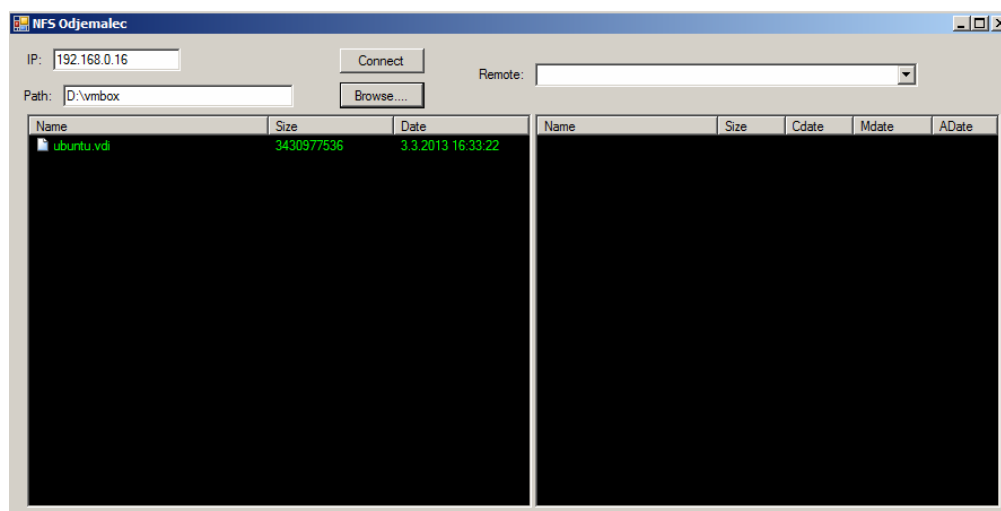
Začeli smo zajem naprave loopback ter v filter nastavili, da je prikazoval samo paketke NFS. Z uporabo ukaza `mount` smo nato ustvarili navidezno povezavo NFS strežnika ter odjemalca, čeprav je vse potekalo na istem operacijskem sistemu. Za zajem vseh potrebnih paketkov smo nato prekopirali datoteke iz `in` v navidezni imenik, ter le tega odprli, da nam je pokazal seznam datotek. Zaradi nadaljnjega testiranja odjemalca NFS v Windows, smo v Oracle VM VirtualBox nastavili virtualno omrežno kartico kot Bridged Adapter.

5.3.2 Priprava programskega okolja

Na računalnik smo namestili Microsoft Visual Studio 2010 ter z njim naredili nov projekt Windows Forms. Namestili smo tudi RemoteTea.NET ter s programom `jrpcgen` ustvarili datoteke za NFS protokol različice 2,3 ter 4.1. Datoteke smo nato vključili v projekt, ter nato začeli z izdelavo osnovnega grafičnega vmesnika.

5.3.3 Izdelava grafičnega vmesnika

V prazno Windows Forms aplikacijo smo dodali polje za naslov NFS strežnika, polje za naslov prikazanega lokalnega imenika ter gumb išči in poveži. Dodali smo kombinirani seznam, kateri bo prikazoval seznam vseh skupnih imenikov v NFS strežniku. Na spodnjo stran aplikacije smo dodali dva seznama datotek ter v nastavitvah pogled nastavili na prikaz podrobnosti. Ustvarili smo refreshLocal funkcijo, katera osveži podan imenik v levi seznam datotek in se izvede ob zagonu aplikacije. Gumbu išči smo nastavili, da prikaže izbor imenika ter z rezultatom pokliče funkcijo refreshLocal. Dodali smo stolpce za dodaten prikaz informacij o datotekah ter dve ikoni za različno vrsto datoteke, katere so imenik ali navadna datoteka.



Slika 16: Prikaz grafičnega vmesnika v zgodnjih fazah izdelave odjemalca NFS

Nadaljnje faze izdelave odjemalca NFS in testiranje so privedle do še večjih sprememb v samem vmesniku ter funkcionalnosti. Dodali smo statusno polje spodaj, ter ustvarili funkcijo, katera v statusno polje zapisuje statusne informacije ter le tega po 2000 vnosih pobriše. Gumb poveži se ob vzpostavitvi povezave s strežnikom preimenuje v prekini povezavo, prav tako se preimenuje naslov aplikacije tako, da poda informacijo o naslovu strežnika ter protokolu s katerim je strežnik povezan. Polja za prikaz datotek smo dodali v panel ter le tega nastavili tako, da se ob povečanju aplikacije povečata tudi seznama datotek in to sorazmerno en do drugega. Dodali smo spodnje gumbe, kateri ob pritisku ali ob kombinaciji tipk naredijo določeno nalogo. Zgoraj smo dodali izvlečni meni ter nova okna nastavitve, pomoč in o izdelku. Dodali smo okno, ki se prikaže ob kopiranju ter premikanju datotek. Prav tako smo izdelali okno, katero se prikaže ob brisanju ali prepisovanju več datotek naenkrat, saj ima primarno okno za povratne informacije premalo izbir. V glavnem oknu smo dodali gumba za dodajanje ter spreminjanje shranjenih strežnikov ter nastavili, da se okno za dodajanje novega strežnika prikaže, če trenutno ni nobenega shranjenega, torej ob prvem zagonu odjemalca.

Polje za naslov NFS strežnika smo odstranili ter nadomestili z kombiniranim seznamom, kateri prikaže seznam vseh že shranjenih strežnikov.

5.3.4 Izdelava NFS knjižnice

Ker smo že v začetku razvoja odjemalca NFS predvidevali dvojni način delovanja, smo se odločili izdelati NFS knjižnico, kot poseben del odjemalca, kateri se lahko uporabi tudi kot knjižnica ali pa vključi v samo aplikacijo. NFS knjižnica vsebuje povezovalne metode za protokole NFS 2, NFS 3 in NFS 4.1. Prav tako smo v NFS knjižnici izdelali določene skupne metode ter strukture. Izdelali smo vmesnik, katere mora vsak protokol podedovati. Metode ki jih vmesnik vsebuje so naslednje:

- **Connect** – procedura ki poveže odjemalec s strežnikom. Atributi kateri morajo biti podani so naslednji: naslov strežnika, uporabniški ID, skupinski ID, kodiranje znakov, uporabi zaščiteni vrata, uporabi predpomnilnik datotečnih identifikatorjev.
- **Disconnect** – procedura ki prekine povezavo s strežnikom. Prekine povezavo ter v primeru NFS 4.1 še pred tem prekine sejo.
- **GetBlockSize** – funkcija katera vrne velikost bloka kateri se bo prenašal med branjem in pisanjem datoteke. V primeru NFS 2 protokola vrne statično 8064, v primeru NFS 3 pokliče proceduro FSINFO, v primeru NFS 4.1 pa vrne rezultat operacije CREATE_SESSION.
- **GetExportedDevices** – funkcija katera vrne seznam skupnih imenikov NFS strežnika. V primeru NFS 4.1 protokola vrne / imenik ter izmenja id odjemalca in ustvari sejo.
- **MountDevice** – procedura katera zavzame podani skupni imenik. Skupni imenik je podan kot atribut procedure. V primeru NFS 4.1 protokola kliče operacijo PUTROOTFH.
- **UnMountDevice** – procedura katera sprosti trenutno zavzeti imenik.
- **GetItemList** – funkcija katera vrne seznam datotek v imeniku kateri je podan kot atribut.
- **GetItemAttributes** – funkcija katera vrne attribute datoteke podane kot atribut.
- **CreateDirectory** – procedura katera ustvari imenik. Atributi so polno ime imenika in s kakšnimi pravicami naj bo imenik ustvarjen.
- **DeleteDirectory** – procedura katera izbriše imenik. Imenik mora biti pred tem izprazen, v nasprotnem primeru procedura vrne napako. V primeru NFS 4.1 procedura kliče DeleteFile, saj je brisanje datoteke enako brisanju imenika.
- **DeleteFile** – procedura katera izbriše datoteko.

- **CreateFile** – ustvari prazno datoteko na NFS strežniku. Podani atributi so polno ime datoteke ter pravice datoteke.
- **Read** – funkcija katera prebere blok iz datoteke. Podani atributi so polno ime datoteke, začetek branja, velikost bloka ter seznam prebranih bajtov. Funkcija vrne rezultat prebranih bajtov iz datoteke.
- **SetFileSize** – procedura katera nastavi novo velikost datoteke. Podani atributi so polno ime datoteke ter nova velikost.
- **Write** – funkcija katera piše v datoteko. Podani atributi so polno ime datoteke, začetek pisanja, velikost bloka ter seznam bajtov kateri bojo napisani v datoteko. Funkcija vrne rezultat napisanih bajtov iz datoteke.
- **Move** – funkcija katera premakne datoteko na NFS strežniku na novo lokacijo. Atributi podani so star imenik, staro ime datoteke, nov imenik ter novo ime datoteke.
- **IsDirectory** – funkcija katera preveri ali je podan naslov imenik. Funkcija kliče GetItemAttributes ter na podlagi zastavice NFDIR = 1 vrne pritrdilni rezultat.
- **CompleteIO** – procedura katero se pokliče na koncu branja ali pisanja v datoteko. Procedura počisti vrednosti trenutno brane ali pisane datoteke in v primeru NFS 4.1 zapre datoteko.

Aplikacija odjemalca kliče te metode, brez da bi se morala ukvarjati s samo strukturo protokola, kateri je v uporabi. Metode so nastale, ko je bil v odjemalec dodan protokol NFS 3.

Za skupno podporo več protokolov naenkrat smo izdelali tudi skupne strukture. Poudarili bomo samo dve, in sicer NFSPermission in NFSAttributes.

NFSPermission je struktura, katera opisuje dovoljen dostop do datoteke ali imenika. Vsebuje dane attribute:

- **_user** – atribut, kateri opiše pravice dostopa do datoteke uporabnika kateri je lastnik
- **_group** – atribut, kateri opiše pravice dostopa do datoteke skupini v kateri je lastnik
- **_other** – atribut, kateri opiše pravice dostopa do datoteke drugim uporabnikom

NFSAttributes je struktura, katera vsebuje attribute o datoteki. Vsebuje dane attribute ter strukture:

- **_cdatetime** – DateTime atribut, kateri poda informacijo kdaj je bil imenik ali datoteka ustvarjena.
- **_adatetime** – DateTime atribut, kateri poda informacijo kdaj je bil imenik ali datoteka nazadnje v uporabi.

- **_mdatetime** – DateTime atribut, kateri poda informacijo kdaj je bil imenik ali datoteka nazadnje v spremenjen.
- **_type** – atribut vrste NFSItemTypes, kateri poda informacijo o tipu datoteke. Ali je datoteka imenik ali je navadna datoteka, ali je simbolična bližnjica ali navadna bližnjica.
- **_mode** – atribut vrste NFSPermission.
- **_size** – velikost datoteke.
- **_handle** – identifikator datoteke, opisan z seznamom bajtov.

5.3.5 Izdelava programskega dela aplikacije

Če izvzamemo NFS knjižnico, se večji del programskega dela aplikacije nahaja v metodah glavnega okna. Najprej smo izdelali proceduro Connect, katera je bila pripeta na istoimenski gumb. Istočasno smo naredili funkciji PingServer ter GetResolvedConnecionIPAddress, kateri pred povezavo preverita ali je vnesen naslov pravilen ter ali se strežnik odziva. Po prvi uspešno opravljeni povezavi z NFS strežnikom smo se lotili seznama skupnih imenikov z uporabo funkcije iz NFS knjižnice. Sam imenik smo na začetku zavzeli z spreminjanjem kombiniranega seznama, šele pozneje smo dodali, da se imenik na prvi poziciji samodejno zavzame ob klicu Connect procedure. Nato smo se lotili izpisa oddaljenega imenika z uporabo procedure RefreshRemote. Samo kopiranje datotek je bilo najprej realizirano z uporabo povleci-spusti akcijskih metod, nato pa smo dodali še akcijo gumbov ter bližnjic. Zaradi zmrzovanja aplikacije med kopiranjem, smo naredili procedure Upload in Download, katere se izvršujejo v svoji niti. Nato smo dodali metode za rekurzivno kopiranje, premikanje ter brisanje datotek in imenikov. Dodali smo podporo za pregled ter urejanje datotek z programom beležnica. Prav tako smo dodali pregled za oddaljene datoteke, le da se ta prvo prekopira v začasni imenik. Za dvojni klik na datoteko v lokalnem seznamu smo naredili, da se izvede primarno dodeljen program. Nazadnje smo dodali podporo, da se skupni imenik iz NFS strežnika preslika v omrežni pogon. To smo dosegli z uporabo knjižnice Dokan[24].

5.3.6 Testiranje aplikacije

Najprej smo testirali povezljivost z NFS strežnikom. Pri tem nam je v veliko pomoč prišla aplikacija Wireshark, saj se je nemalokrat zgodilo, da smo prav zaradi te aplikacije odkrili napako ter le to popravili. Kot primer naj navedemo slabo kodirano strukturo v XDR formatu, za katero je Wireshark znal pokazal, kje je napaka. Testirali smo prenos datoteke ter pisanje v strežnik in vse metode ki jih NFS knjižnica podpira.

Nadaljnji test prikaza oddaljenega imenika z velikim številom datotek je pokazal, da je potrebna optimizacija odjemalca. Najprej smo optimizirali razvrščanje seznama datotek tako, da se je seznam razvrščal le enkrat, zatem smo optimizirali pridobivanje atributov ter obveščali uporabnika o stanju napredka. Dodali smo opcijo shranjevanja atributov ter identifikatorjev, kar je še posebej pospešilo izpis velika števila datotek v imeniku.

Hitrost prenosa datoteke se je povečal, če smo povečali velikost bloka, kateri se je prenašal preko READ ali WRITE procedur in operacij. Zato smo na začetku povezave z NFS strežnikom le tega povprašali o največji dovoljeni hitrosti.

Odjemalec smo testirali tudi na način, da smo ga pustili neaktivnega veliko časa ter potem hoteli pridobiti vrednost imenika ali datoteke. Izkazalo se je da v primeru NFS 4.1 seja poteče ter vrne napako BAD_SESSION. Šele po nadaljnjem testiranju v Ubuntu operacijskem sistemu smo ugotovili kolikšen čas je potreben, da seja ne zastara.

5.4 Problemi tekom izdelave

Tekom izdelave odjemalca NFS so se pojavili številni problemi. Izpostavili bomo tri največje probleme, kateri so se nam pojavili ob izdelavi.

Prvi večji problem se je pojavil ob samem povezovanju v NFS strežnik. NFS strežnik je odgovoril na list vseh skupnih imenikov, ko smo pa zavzeli dani imenik je NFS strežnik odgovoril s ERR_ACCESS. Pomislili smo, da je problem identifikator uporabnika ter identifikator skupine in le te nastavili na skrbnika sistema ter na novo kreiranega uporabnika. Odgovor je bil zmeraj isti. Zamenjali smo tudi transportni protokol, vendar je bil rezultat isti. Podrobno smo analizirali povezavo med odjemalcem in strežnikom na Ubuntu sistemu, vendar nismo dobili konkretnega odgovora, kaj bi lahko bilo narobe. Po nadaljnjem iskanju na spletu se je pojavila informacija, da so mogoče kriva vrata izvora odjemalca. Namreč nekateri NFS strežniki zavračajo povezave, če so vrata odjemalca manjša od 665 ali večja od 1023. Ta vrata so nepriviligirana oziroma negotova. Razlog za to tiči v večini operacijskih sistemih, saj lahko strežnik na teh vratih postavi vsak uporabnik, na privilegiranih vratih pa samo skrbnik sistema. Da je naš odjemalec uporabljal točno določena vrata izvora, smo morali narediti spremembo v RemoteTea.Net povezovalnih metodah, in sicer v OncRpcTpcClient in OncRpcUdpClient. Ustvarili smo funkcijo GetLocalPort, katera vrne naključna nezasedena vrata med števili 665 in 1023. Funkcija je predstavljena v sliki 17.

```

private int GetLocalPort()
{
    Random random = new Random();
    int randomNumber = 0;
    bool inUse = false;
    do
    {
        randomNumber = random.Next(665, 1023);
        inUse = false;
        foreach (var tcpListener in IPGlobalProperties.GetIPGlobalProperties().GetActiveTcpListeners())
        {
            if (tcpListener.Port == randomNumber)
            {
                inUse = true;
                break;
            }
        }
    } while (inUse);
    return randomNumber;
}

```

Slika 17: Primer generatorja naključnih TCP prostih vrat odjemalca

Te vrata smo uporabili pred samo prvo povezavo s NFS strežnikom in določitvijo njegovega naslova, ter samo če smo to nastavili v nastavitvah odjemalca.

```

if (useSecurePort)
{
    var localEp = new IPEndPoint(IPAddress.Any, GetLocalPort());
    socket.Bind(localEp);
}

```

Po tej modifikaciji se nam je odjemalec uspešno povezal na NFS strežnik.

Drugi večji problem na katerega smo naleteli, je bil pri branju ali pisanju datoteke. V dokumentaciji[14] je pisalo naslednje:

```

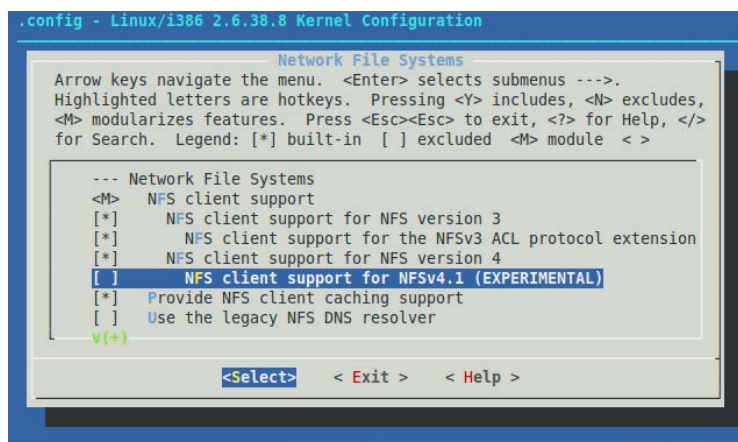
/*
 * The maximum number of bytes of data in a READ or WRITE
 * request.
 */
const MAXDATA = 8192;

```

Če smo velikost bloka nastavili na 8192, se nam datoteka ni pravilno prenesla, če smo pa velikost nastavili na 4096, je prenos potekal pravilno. Problem je nastal, ker smo hoteli imeti kar se da velik blok, še posebej pri protokolu NFS 2, kateri je zgoraj omejen na 8192 bajtov. S poskusom ter analizo smo ugotovili da je največja velikost bloka na NFS 2 protokolu 8064 bajtov. Dokumentacija je torej zavajala za dani primer, oziroma smo morali upoštevati velikost celotnega zahtevka, ki je bil poslan s strani odjemalca. Vrednost na NFS 3 pa je bila 8000 bajtov oziroma 192 manj od predvidene vrednosti zapisane v FSINFO. Vrednost v NFS 4.1 pa je bila 7900 bajtov oziroma 292 manj od predvidene zapisane v CREATE_SESSION operaciji. Težava je bila predvsem v tem, da strežnik ni vrnil nobene napake, samo datoteka

se ni pravilno prenesla, zaradi česar je bilo testiranje zelo dolgotrajno. Po nadaljnjem testiranju smo ugotovili, da je vrednost, ki jo odštejemo od maksimalno velikega bloka velikost glave zahtevka.

Tretji večji problem pa se je pojavil pri sami analizi protokola NFS 4.1. Ubuntu operacijski sistem ima namreč strežnik NFS 4.1 na voljo kot namestitveni paket, sam odjemalec pa je vključen v samo jedro operacijskega sistema. Veliko časa smo porabili, da smo imeli delujoč odjemalec NFS 4.1. Najprej smo mislili, da je problem star operacijski sistem in smo le tega posodobili na najnovejšo različico. Izkazalo se je, da tudi najnovejša različica ne podpira odjemalca in tudi podpora strežniku NFS 4.1 je izginila. Namestili smo tudi drugo distribucijo operacijskega sistema Linux, a tudi ta ni imela omenjene podpore. Zato smo se odločili, da se vrnemo na začetno različico Ubuntu ter v tej omogočimo odjemalec NFS 4.1 tako, da na novo prevedemo jedro z nastavitvijo podpore NFS 4.1 odjemalcu. Slika 18 prikazuje kako omogočimo odjemalec NFS 4.1 v nastavitvah jedra Linux z ukazom `make menuconfig`. Kot vidimo, je odjemalec v preizkusni dobi in zato tudi ni primarno omogočen.



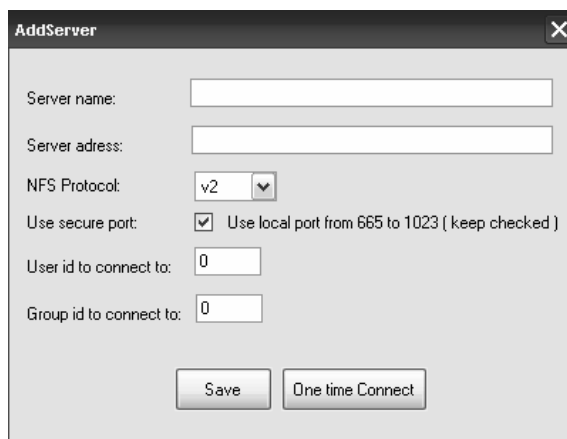
Slika 18: Omogočanje odjemalca NFS 4.1 v jedru Linux

Po prevedenemu jedru je odjemalec NFS 4.1 deloval in lahko smo začeli analizirati sam protokol.

5.5 Predstavitev aplikacije

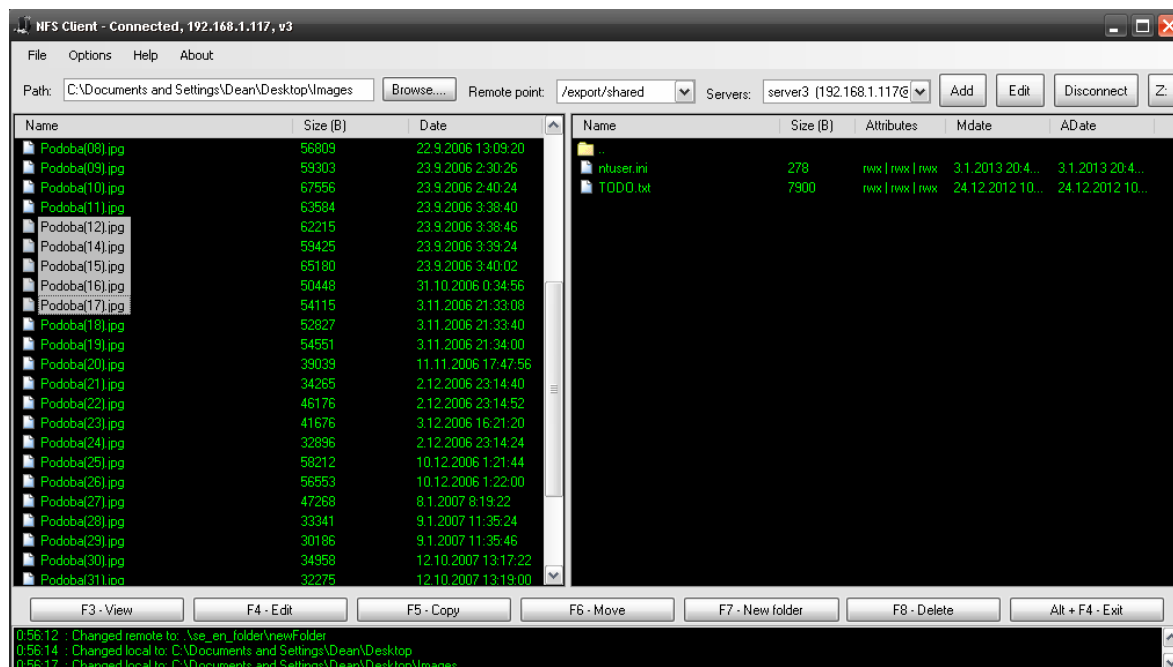
V tem poglavju bomo podrobno predstavili samo aplikacijo odjemalec NFS. Odjemalec NFS za delovanje potrebuje .NET Framework 4.0 ter knjižnico Dokan. Obe knjižnici sta priloženi v instalacijski datoteki odjemalca NFS.

Ko prvič zaženemo Odjemalec NFS, se nam čez glavno okno prikaže okno za dodajanje strežnika prikazano v sliki 19. V oknu strežnik poimenujemo ter napišemo naslov strežnika. Izberemo NFS protokol, s katerim se želimo povezati ter po želji nastavimo uporabniški identifikator ter skupinski identifikator. Strežnik shranimo ali pa se odločimo da se bomo na ta strežnik povezali samo enkrat. V tem primeru se strežnik ne shrani, ampak avtomatično poveže.



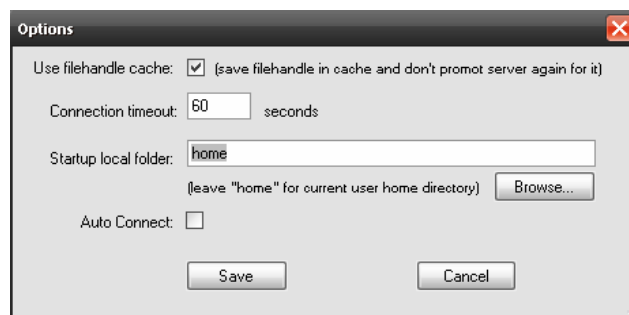
Slika 19: Prikaz okna za dodajanje novega strežnika

Ko dodamo strežnik, se nam prikaže glavno okno odjemalca. Slika 20 prikazuje glavno okno odjemalca, ko je ta že povezan z NFS strežnikom na naslovu 191.168.1.117.



Slika 20: Prikaz glavnega okna aplikacije

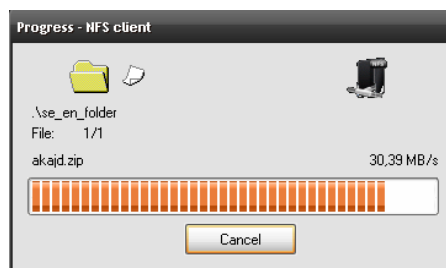
Zgoraj levo je prikazana trenutna pot imenika prikazana v lokalnem seznamu datotek, desno pa je prikazana trenutno izbrana skupna mapa NFS strežnika. Desno je prikazan seznam shranjenih strežnikov, ki prikazuje trenutno izbranega strežnika. Nato sta zraven gumba, katera sta namenjena za dodajanje novih strežnikov ali spreminjanje obstoječih. S spreminjanjem obstoječih strežnikov se pojavi podobno okno, kot je prikazano na sliki 18, vendar z razliko, da je namesto gumba za enkratno povezavo tam gumb za brisanje strežnika. Levo okno prikazuje lokalni imenik, desno pa imenik na NFS strežniku. Spodaj imamo gumbe za predogled datoteke, za urejanje datoteke, za kopiranje datotek, za premikanje datotek, za kreiranje novega imenika, za brisanje datotek ter za izhod iz aplikacije. Na dnu aplikacije se nahaja statusni seznam, kateri časovno prikazuje pretekle operacije, ki jih je odjemalec naredil. Zgornji meni ima možnosti za povezovanje ter izhod iz aplikacije ter prikaz nastavitve aplikacije, pomoč o odjemalcu in o odjemalcu. Slika 21 prikazuje okno z nastavitvami aplikacije.



Slika 21: Prikaz okna z nastavitvami

Same nastavitve aplikacije so trenutno samo štiri, vendar se bodo z nadaljnjim razvojem aplikacije pojavile še nove. Prva nastavitev omogoča hranjenje atributov v hitrem seznamu. To zelo pohitri prebiranje oddaljenih imenikov, saj se potem ko je enkrat imenik prebran, atributi ne prebirajo več. Seznam datotek v imeniku se še zmeraj prebere, tako da če je bila datoteka na strežniku izbrisana, se le ta ne prikaže. Naslednja nastavitev je število sekund od poslanega paketa odjemalca do preteka čakanja odgovora. Če v tem času odjemalec ne dobi odgovora, se povezava prekine. Naslednja nastavitev je prikazan imenik ob zagonu aplikacije. Če je vrednost nastavljena na `home`, se bo prikazala vsebina domačega imenika uporabnika. V nasprotnem primeru bo prikazan imenik, katerega vnesemo v polje za vnos ali poiščemo z uporabo gumba. Zadnja nastavitev je opcija, ali se odjemalec ob zagonu samodejno poveže v strežnik, kateri je bil nazadnje uporabljen.

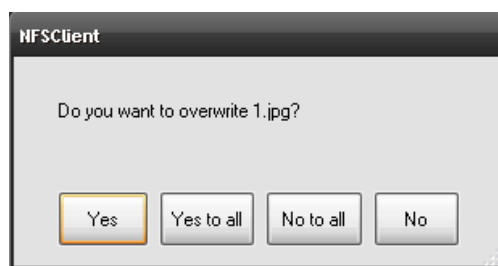
Ob kopiranju ali premikanju datotek pa se nam pojavi še okno, katero prikazuje stanje o napredku. Prikazano je v sliki 22.



Slika 22: Prikaz okna za kopiranje datotek

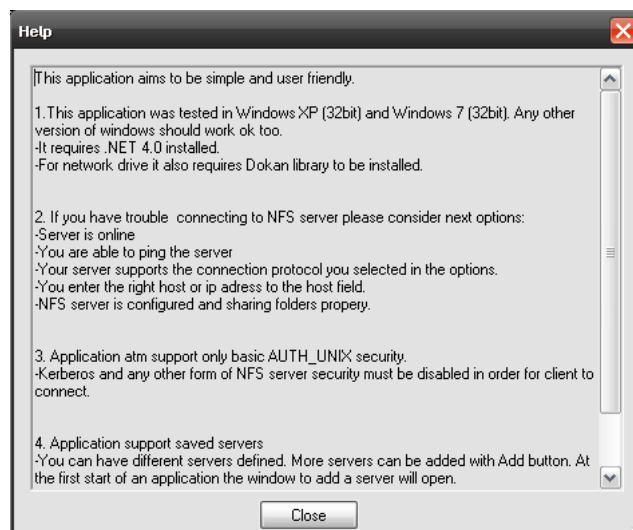
V oknu je na vrhu prikazana animacija kopiranja datoteke. Pod njo se nam pojavi ime imenika, v katerega se trenutno kopira datoteka. Nato še informacija, katera datoteka v tem imeniku se kopira od vseh izbranih datotek. Spodaj je prikazano še ime datoteke ter napredek. Če hočemo prekiniti prenos, pritisnemo spodnji gumb. Ko je prenos končan, se nam osveži imenik, v katerega je bila datoteka prenesena. V primeru, da datoteko premikamo, se nam osveži lokalni imenik kot oddaljeni imenik.

Če datoteka že obstaja, se pojavi okno, ki je prikazano na sliki 23. Omogoča štiri izbire in sicer prepisi datoteko, prepisi vse datoteke, ne prepisi nobene datoteke ter ne prepisi trenutne datoteke.



Slika 23: Prikaz okna za prepisovanje obstoječih datotek

V aplikaciji je tudi okno za prikaz pomoči. Okno izpiše vsebino tekstovnega polja in ponuja osnovne podatke o aplikaciji ter pomoč ob problemu z vzpostavitevjo povezave z strežnikom. Okno se bo nadgradilo z dodatnimi temami ter razlagami o problemu z uporabo aplikacije. Prikazano je v sliki 24.



Slika 24: Prikaz okna za pomoč

5.6 Objava izvirne kode na spletu

Zaradi nadaljnjega razvijanja aplikacije tudi s pomočjo drugih razvijalcev, smo se odločili objaviti izvirno kodo na spletu. Za objavo smo uporabili spletni servis GitHub[25].

Najprej smo naredili račun za spletnih servis GitHub in v njem ustvarili projekt NFSCClient. Naložili smo si odjemalec za operacijski sistem Windows, kateri poenostavi dodajanje in pregled datotek v projektih, katere imamo na voljo. S pomočjo odjemalca smo prenesli datoteke v spletni servis. Prav tako smo ustvarili datoteko README.md, katera poda opis projekta v spletnem servisu. V njega smo vnesli osnovne podatke o aplikaciji ter dodali sliko glavnega okna aplikacije. Nadaljnje spremembe v aplikaciji smo zmeraj vnašali preko odjemalca GitHub in tako imeli vedno na razpolago pregled nad spremembami prejšnje prenesene različice aplikacije. Ustvarili smo tudi datoteko LICENCE, katera vsebuje licenco pod katero je aplikacija razvita.

Za licenco smo izbrali GPL različico 3, katera je najnovejša licenca GPL. GPL licenca je najbolj razširjena licenca pri odprto kodnih programih. Omogoča končnim uporabnikom svobodo do uporabe, preučevanja, deljenja in spreminjanja programske kode. Programska koda razvita s pomočjo licence GPL je brezplačna in vedno prosto dostopna.

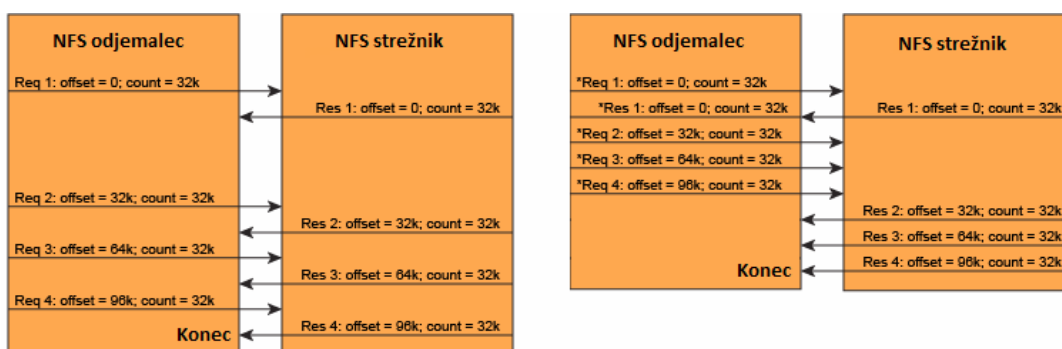
Izvorna koda aplikacije je na voljo na spletnem naslovu:

<https://github.com/DeCoRawr/NFSCClient>

6 Sklepne ugotovitve

Med razvojem aplikacije smo naleteli na težave, katere smo uspešno odpravili ter razvili delujoč odjemalec NFS opisan v poglavju 5.5. Veliko časa smo posvetili sami analizi NFS protokola ter pripadajoči dokumentaciji. Že iz samega razloga, da je protokol NFS razmeroma star, ter za njega ni brezplačnega odjemalca Windows, je razvidno, da izdelava odjemalca NFS ni bila enostavna. Razvoj aplikacije je potekal veliko časa in ima še vedno veliko pomanjkljivosti ter možnosti nadgradenj. Nekaj planiranih nadgradenj:

- Podpora branju vnaprej (angl. read-ahead) in pisanju vnaprej (angl. write-behind). Ta podpora omogoča še hitrejši prenos datotek med odjemalcem in strežnikom. Slika 25 prikazuje delovanje branje vnaprej. Podaja se zahtevo po prvem bloku datoteke, ko je ta potrjena, se namesto naslednje poda zahtevo po več blokih naenkrat, brez potrebe, da bi čakali na odgovor naslednje. Tako se pospeši hitrost prenosa, saj se hkrati prejema več blokov naenkrat. Prav tako kot branje vnaprej, pisanje vnaprej po prejetju prvega odgovora poda zahtevo po več pisanj hkrati.



Slika 25: Prikaz branja vnaprej v NFS protokolu

- Podpora varnostnim algoritmom. Odjemalec trenutno podpira samo AUTH_UNIX zaščito, katera pa ni varna, saj je podan samo identifikator uporabnika ter identifikator skupine. Ker povezava lahko ni zaščitena, se lahko s prisluškovanjem le te pridobi ter ponaredi.
- Podpora pNFS opisanemu v poglavju 4.7.1. Trenutno je podprt NFS 4.1 protokol, vendar brez podpore pNFS.
- Podpora ukaznim argumentom. S pomočjo ukaznih argumentov se lahko odjemalec avtomatično poveže na dani NFS strežnik ter preslika v omrežni pogon že ob prijavi v

Windows sistem. Podpira naj več strežnikov naenkrat. Primer: `NFSClient.exe -a 192.168.0.13 -v 4.1 -m Z:`

- Podpora predogledu datotek, brez da so le te zapisane na datotečni sistem odjemalca.
- Podpora nadaljnjemu protokolu NFS 4.2 opisanemu v poglavju 4.8.

Tudi s številnimi možnostmi nadgradenj odjemalec izpolnjuje osnovne ter vse dodatne zahteve, katere so bile zadane pred začetkom izdelave. Prav tako je odjemalec dobro testiran ter pripravljen za množično uporabo. Potrudili se bomo, da bo odjemalec na voljo vsem uporabnikom spleta ter da bo le ta prosto dostopen.

Literatura

- [1] (2013) Wikipedia, Microsoft Visual studio. Dostopno na:
http://en.wikipedia.org/wiki/Microsoft_Visual_Studio
- [2] (2013) RemoteTea.net project page. Dostopno na:
<http://remoteteanet.sourceforge.net/>
- [3] (2013) Wikipedia, Wireshark. Dostopno na:
<http://en.wikipedia.org/wiki/Wireshark>
- [4] (2013) Wikipedia, VirtualBox. Dostopno na:
http://en.wikipedia.org/wiki/Virtual_box
- [5] B. Callaghan, NFS Illustrated, Boston: Addison-Wesley Professional, 1999
- [6] (2013) Wikipedia, Network File System. Dostopno na:
http://en.wikipedia.org/wiki/Network_File_System
- [7] (2013) Wikipedia, Remote procedure call. Dostopno na:
http://en.wikipedia.org/wiki/Remote_procedure_call
- [8] (1988) IETF, RFC 1050 - Remote Procedure Call Protocol Specification. Dostopno na:
<http://tools.ietf.org/html/rfc1050>
- [9] (2009) IETF, RFC 5531 - Remote Procedure Call Protocol Specification Version 2.
Dostopno na: <http://tools.ietf.org/html/rfc5531>
- [10] (2013) Wikipedia, External Data Representation Dostopno na:
http://en.wikipedia.org/wiki/External_Data_Representation
- [11] (1987) IETF, RFC 1014 - External Data Representation Standard. Dostopno na:
<http://tools.ietf.org/html/rfc1014>
- [12] (1995) IETF, RFC 1832 - External Data Representation Standard. Dostopno na:
<http://tools.ietf.org/html/rfc1832>
- [13] (2006) IETF, RFC 4506 - External Data Representation Standard. Dostopno na:

<http://tools.ietf.org/html/rfc4506>

[14] (1989) IETF, RFC 1094 - Network File System Protocol Specification. Dostopno na:
<http://tools.ietf.org/html/rfc1094>

[15] (2000) IETF, RFC 3010 - NFS version 4 Protocol. Dostopno na:
<http://tools.ietf.org/html/rfc3010>

[16] (2003) IETF, RFC 3530 - Network File System (NFS) version 4 Protocol. Dostopno na:
<http://tools.ietf.org/html/rfc3530>

[17] (2010) IETF, RFC 5661 - Network File System (NFS) Version 4 Minor Version 1
Protocol. Dostopno na: <http://tools.ietf.org/html/rfc5661>

[18] (1995) IETF, RFC 1813 - NFS Version 3 Protocol Specification. Dostopno na:
<http://tools.ietf.org/html/rfc1813>

[19] (1996) IETF, RFC 2051 - WebNFS Client Specification. Dostopno na:
<http://tools.ietf.org/html/rfc2054>

[20] (2012) IETF, draft-ietf-nfsv4-minorversion2-17. Dostopno na:
<http://tools.ietf.org/html/draft-ietf-nfsv4-minorversion2-17>

[21] (2013) Wikipedia, Portmap. Dostopno na:
<http://en.wikipedia.org/wiki/Portmap>

[22] (1995) IETF, RFC 1833 - Binding Protocols for ONC RPC Version 2 Dostopno na:
<http://tools.ietf.org/html/rfc1833>

[23] (2013) SourceForge, Remote Tea Dostopno na:
<http://sourceforge.net/apps/trac/remotetea/wiki/WikiStart>

[24] (2013) Dokan Dostopno na:
<http://dokan-dev.net/en/>

[25] (2013) Wikipedia, GitHub Dostopno na:
<http://en.wikipedia.org/wiki/GitHub>