FAKULTET ELEKTROTEHNIKE I RAČUNARSTVA

Network File System

(NFS)

Sustavi za praćenje i vođenje procesa

ZAGREB, Lipanj 2005

Saša Janjić

0036385407

Sadržaj

Uvod

Mrežni Datotečni Sustav (NFS – Network File System) pruža transparentan udaljeni pristup dijeljenim resursima na mreži. NFS protokol stvoren je s ciljem da bude prijenosan na različite platforme, operativne sustave, mrežne arhitekture, i transportne protokole. Portabilnost je postignuta korištenjem RPC (Remote Procedure Call) primitiva koje se nalaze povrh XDR (eXternal dana Representation) sloja. Implementacije postoje za mnoge platforme, od osobnih do superračunala.

Pregled, povijest i standardi

Povijesti Interneta i TCP/IP protokola imaju zajedničke korijene. Međutim postoji i, mnogo manje spominjana, ali sa velikom zaslugom u povijesti razvoja navedenih tehnologija, treća karika. To je operativni sustav koji se vrtio na računalima za vrijeme ranog Interneta, a koji se još i danas koristi na mnogim Internet serverima diljem svijeta: UNIX operativni sustav.

Sun Microsystems je jedan od pionira u razvoju UNIX-a, kao i TCP/IP mreža. U začecima TCP/IP-a, stvoreni su određeni alati da bi se omogućio pristup drugim računalima putem mreže. Protokoli za udaljeni pristup poput Telneta omogućili su logiranje korisnika na drugo računalo i korištenje njegovih resursa. FTP (File Transfer Protocol) uveo je mogućnost kopiranja datoteka sa tuđeg na vlastito računalo.

Međutim, niti jedno od navedenih rješenja ne pruža pristup datotekama na udaljenom računalu na način sličan pristupu datotekama na lokalnom računalu. Kao odgovor na tu potrebu, Sun je napravio Network File System (NFS). NFS je osmišljen specifično sa ciljem uklanjanja razlike u radu s lokalnim i udaljenim datotekama. Za korisnika, nakon odgovarajućeg podešavanja, datoteka na udaljenom računalu može se koristiti kao da se nalazi na disku korisnikovog lokalnog stroja. Sun je NFS osmislio da bude neovisan o proizvođaču; da bi osigurao da bi sklopovlje koje je napravio Sun te ono drugih proizvođača bilo kompatibilno.

Osnove NFS arhitekture i opće operacije

NFS prati klasičan TCP/IP klijent/server model rada. Tvrdi disk ili direktorij na uređaju za pohranu nekog računala može se podesiti da bude dijeljeni resurs. Tom resursu mogu pristupati klijenti koji montiraju dijeljeni direktorij ili uređaj, tako da se on ponaša kao lokalni direktorij na računalu klijenta. Neka računala mogu biti klijenti ili poslužitelji, dok neka mogu biti oboje: djeleći neke svoje i pristupajući resursima koje pružaju ostali.

NFS arhitektura uključuje tri glavne komponente koje definiraju njegovo djelovanje. *XDR* (eXternal Data Representation) standard definira prikaz podataka u razmjenama između klijenta i poslužitelja. *RPC* (Remote Procedure Call) protokol koristi se kao metoda pozivanja procedura na udaljenom računalu. NFS procedure i operacije rade korištenjem RPC-a za obavljanje raličitih zahtjeva. Zaseban Mount protokol koristi se za montiranje resursa kao što je gore navedeno.

Jedan od važnijih zahtijeva pri stvaranju NFS-a bile su performanse. Očito je da čak i ako je datoteka na udaljenom računalu podešena tako da djeluje kao lokalna, stvarne operacije čitanja i pisanja putuju kroz mrežu. To obično zahtijeva više vremena nego obično slanje podataka unutar računala, pa je protokol trebao biti što robusniji. Takav način razmišljanja rezultirao je nekim, na prvi pogled, iznenađujućim odlukama, poput korištenja nepouzdanog datagramskog prijenosa (UDP) za prijenos u TCP/IP-u, umjesto pouzdanog TCP-a koje koristi većina protokola za prijenos. To ima zanimljive posljedice na rad samog protokola.

Još jedna o ključnih točaka NFS-a je jednostavnost. Za NFS poslužitelje kaže se da su *stateless*, što

znači da je protokol napravljen sa ciljem da poslužitelj ne mora voditi računa o tome koje datoteke su otvorene te koji su ih korisnici otvorili. Tako su zahtijevi međusobno nezavisni i omogućuju poslužitelju da se elegantno nosi sa događajima poput padova sustava, bez potrebe za složenim postupcima oporavka. Dizajn protokola osigurava izbjegavanje konflikata u radu s datotekama u slučaju gubitka ili dupliciranja zahtijeva.

Inačice i standardi

Od svog nastanka pa do danas, NFS je postao de facto standard. Prva i još uvijek korištena inačica NFS protokola je verzija 2. Ona je svojedobno uvedena kao službeni TCP/IP standard izdavanjem *RFC-a 1094, NFS: Network File System Protocol Specification*, 1989 godine.

Inačica 3 NFS protokola objavljena je 1995 godine kao *RFC 1813, NFS Version 3 Protocol Specification*. Slična je inačici 2 ali uz nekoliko izmjena i novih mogućnosti. Između ostalog sadrži podršku za prijenos većih datoteka, bolju podršku za podešavanje atributa datoteka, i nekoliko novih procedura za pristup datotekama i manipulaciju. Inačica 3 također pruža podršku za veće datoteke od inačice 2.

Naredna, inačica 4 NFS standarda objavljena je 2000 godine kao *RFC 3010, NFS version 4 protocol*. Za razliku od prethodne koja je donijela samo manje promjene, NFSv4 je skoro pa nanovo napisan NFS protokol. Uključuje brojne promjene, od kojih su najvažnije:

- U skladu s modernim radom u Internet okružju, NFSv4 stavlja veći naglasak na sigurnost
- NFSv4 uvodi koncept *Compuond* procedura, koje pružaju mogućnost slanja više jednostavnih procedura od strane klijenta prema poslužitelju, kao grupe
- NFSv4 gotovo podvostručuje broj pojedinačnih procedura koje su klijentu na raspolaganju pri pristupu datotekama na NFS poslužitelju
- Inačica 4 ujedno unosi i značajnu novost u slanje poruka, postavljanjem TCP-a kao transportnog protokola za NFSv2
- Konačno, funkcije Mount protokola integrirane su u osnovni NFS protokol, uklanjajući time potrebu za zasebnim protokolom kao što je to slučaj u prethodnim inačicama

RFC 3010 kasnije je, u Travnju 2003. zamijenjen sa *RFC 3530, Network File System (NFS) version 4 Protocol*. Taj standard donosi nekoliko revizija i poboljšanja u radu NFS verzije 4.

NFS arhitektura i komponente

Sa stanovišta TCP/IP protokola kao cjeline, NFS je jedinstven protokol koji se nalazi u aplikacijskom sloju TCP/IP modela. Taj TCP/IP sloj uključuje sesijski, prezentacijski te aplikacijski sloj OSI referentnog modela. Iako je razdvajanje posljednja 3 sloja OSI modela ponekad suvišno, postoje slučajevi kada se oni pokazuju korisni u shvaćanju arhitekture protokola, a to je slučaj i sa NFS-om.

Arhitektura i glavne komponente

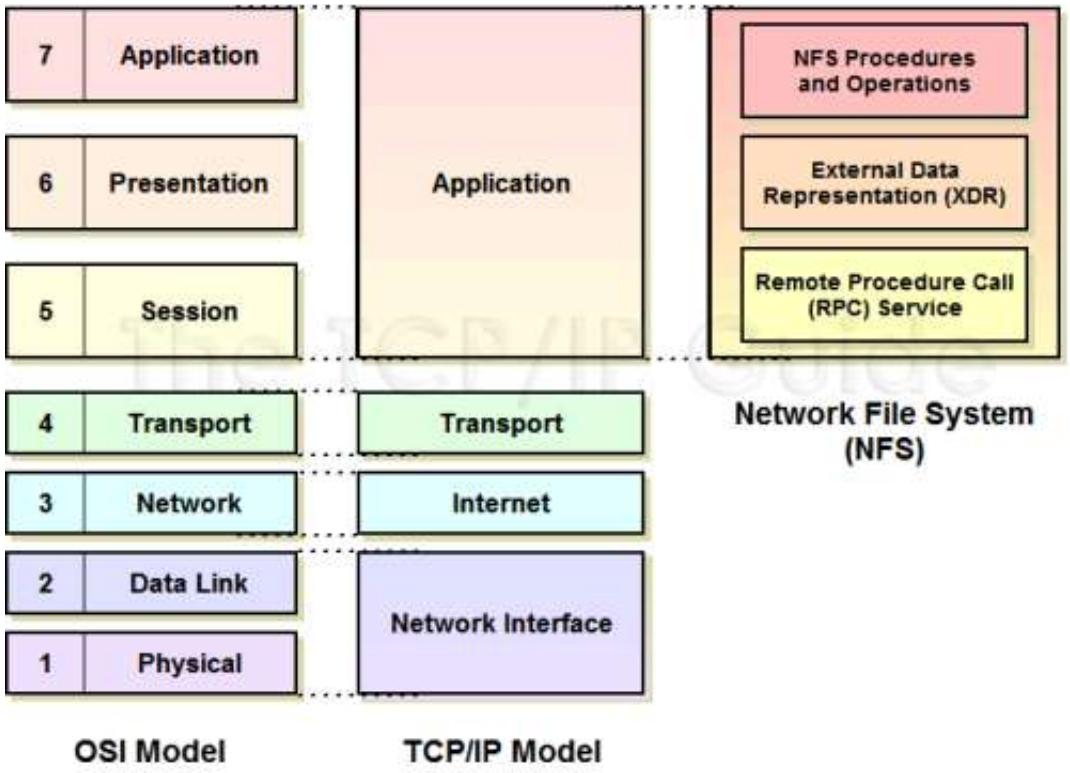
Djelovanje NFS-a određeno je trima glavnim komponentama koje se mogu promatrati kao da logički leže u pojedinim od tri sloja OSI modela odgovarajući pritom TCP/IP aplikacijskom sloju. Te komponente su:

Remote Procedure Call (RPC): RPC je generička usluga sesijskog sloja kojom se implementira klijent/poslužitelj funkcionalnost povezivanja mreža. Ona proširuje pojam programskog pozivanja lokalne procedure na određenom računalu, na pozivanje procedure na udaljenom računalu putem mreže.

External dana Representation (XDR): XDR je opisni jezik koji omogućuje definiranje tipova podataka na odgovarajući način. XDR konceptualno leži na prezentacijskom sloju; njegova jedinstvena reprezentacija omogućuje razmjenu podataka NFS-om među računalima koja možda koriste različite interne metode pohrane podataka.

NFS procedure i operacije: Stvarna funkcionalnost NFS-a implementirana je u obliku procedura i operacija koje konceptualno djeluju na sedmom sloju OSI modela. Te funkcije određuju zadatke koji će biti izvršeni nad datotekama na mreži, koristeći XDR za prikaz podataka te RPC za izvršenje naredbi na povezanoj mreži.

Ova tri ključna potprotokola čine glavninu NFS protokola. Svaki je pojedinačno opisan detaljnije u zasebnom poglavlju ovog dijela o NFS-u.



Ilustracija 1NFS komponente

NFS se arhitekturalno nalazi u TCP/IP aplikacijskom sloju. Iako se u TCP/IP modelu ne radi stroga razlika između funkcija slojeva od pet do sedam OSI referentnog modela, tri NFS potprotokola im dobro odgovaraju.

Ostale važne funkcije

Uz navedene tri komponente, NFS protokol kao cjelina uključje još nekoliko funkcija, od kojih je neke korisno spomenuti:

- **Mount protokol:** Tvorci NFS standarda odlučili su da se NFS neće baviti pojedinačnim otvaranjem i zatvaranjem datoteke. Umjesto toga, zaseban protokol se koristi u tu svrhu. Pristup

datoteci ili drugom resursu na mreži zahtijeva njegovo prvotno montiranje korištenjem navedenog protokola. Mount protokol je arhitekturalno zaseban, ali očito blisko vezan uz NFS, pa je čak i definiran u poglavlju NFS standarda. Opisan je u zadnjem dijela ovog poglavlja. (U NFSv4 funkcije Mount protokola integrirane su u sam NFS protokol)

- **NFS model datotečnog sustava:** NFS koristi određeni model za implementiranje datotečne i direktorijske strukture sustava koji ga koristi. Taj je model blisko baziran na modelu datotečnog sustava UNIX-a, ali nije strogo vezan uz taj operativni sustav.
- **Sigurnost:** Inačice 2 i 3 uključuju samo ograničen skup sigurnosnih odredbi. Koriste UNIX način autentikacije za provjeru dozvole pri različitim operacijama. NFSv4 značajno popravlja postojeće sigurnosne opcije NFS implementacije. To uključuje opciju višestruke autentikacije i enkripcijske algoritme, a načinjene su mnoge promjene u samom radu protokola.

Poput ostalih TCP/IP protokola, NFS je ostvaren u obliku klijentskog i poslužiteljskog software-a koji implementira gore navedene funkcije. NFS standardi, osobito inačice 3 i 4, bave se brojnim stvarima vezanim uz ispravnu NFS klijentsko-poslužiteljsku implementaciju, uključujući interakciju između klijenta i servera, zaključavanje datoteka, dozvole pristupa, caching, pravila retransmisije, međunarodnu potporu i još mnogo toga.

Pohrana i tipovi podataka, i XDR (eXternal dana Representation) standard

Glavna ideja iza NFS standarda je dopustiti nekome za jednim računalom da čita ili piše u datoteku na drugom računalu jednako kao što to čini na vlastitom stroju. Naravno, datoteke na lokalnom računalu pohranjene su na istom datotečnom sustavu, uz korištenje iste strukture podataka i uz isti smisao predstavljanja različitih tipova podataka. To ne mora nužno vrijediti za uređaj kojem se udaljeno pristupa, i to predstavlja problem koji NFS mora riješiti.

Metode za jedinstvenu razmjenu podataka: XDR

Da bi se osigurala vjerodostojnost prikaza, jedan od pristupa je ograničenje pristupa na samo one udaljene datoteke koje se nalaze na računalima sa istim operacijskim sustavom. Takav bi, međutim, postupak višestruko smanjio učinkovitost NFS-a. Također ne bi bilo praktično zahtijevati da svako računalo razumije internu reprezentaciju svakog drugog računala. Potrebna je općenitija metoda koja bi i različitim platformama dopustila dijeljenje podataka. Tvorci NFS-a odlučili su da on barata podacima korištenjem jedinstvenog jezika za opis podataka. Taj jezik nazvan je *External data Representation* (XDR) i izvorno je opisan u RFC 1014; kasnije nadograđen u *RFC 1832, XDR: External dana Representation Standard*, 1995.

XDR radi na sljedeći način. Kada se informacija o tome kako pristupiti datoteci treba prenijeti od uređaja A do uređaja B, uređaj A ju prvo prevodi iz svog internog prikaza u XDR prikaz. Informacija se prenosi mrežom korištenjem XDR enkodiranja. Tada, uređaj B tu informaciju prevodi sa XDR u svoj vlastiti interni prikaz, tako da je ona korisniku prikazana kao da se nalazi na lokalnom datotečnom sustavu. Svaki uređaj treba znati samo kako prevesti sa svog "jezika" u XDR prikaz i obrnuto; uređaji A i B ne moraju ništa znati o međusobnim internim detaljima. Ovakvo prevođenje je naravno uobičajen posao prezentacijskog sloja, gdje XDR leži u OSI referentnom modelu. Sam XDR temelji se na ISO standardu pod nazivom ASN (Abstract Syntax Notation).

Opisana ideja koristi se i u drugim protokolima za razmjenu podataka neovisno o vrsti pripadnih sustava. Na sličnoj ideji temelji se rukovanje informacijama u razmjeni korištenjem SNMP (Simple Network Management Protocol) protokola. Isti princip nalazi se i u pozadini NVT (Network Virutal

Terminal) paradigmi korištenoj u Telnet protokolu.

XDR tipovi podataka

Da bi XDR bio jedinstven, mora omogućiti opis uobičajenih tipova podataka korištenih u računalu. Na primjer, mora omogućiti razmjenu podataka kao što su cijeli brojevi (integers), brojevi s pomičnim zarezom, nizovi, te ostale. XDR standard opisuje strukturu podataka korištenjem notacije vrlo slične jeziku "C". To je jedan od najpoznatijih programskih jezika u računalnoj povijesti i usko je vezan uz UNIX, a time i uz određene TCP/IP tehnologije.

XDR pruža mogućnost definiranja novih tipova podataka i metodu za određivanje opcionalnih podataka. To, uz velik broj podržanih tipova podataka, još više doprinosi fleksibilnosti.

U tablici 1 prikazani su tipovi podataka definirani XDR-om, koje NFS može koristiti u razmjeni podataka između klijenta i poslužitelja. U tablici su uključeni kod tipa, njegova veličina u bajtovima, ime te opis.

Tablica 1: XDR tipovi podataka

Kod tipa podatka	Veličina (byte)	Opis
int	4	**Cjelobrojni tip s predznakom:** 32-bitni cijeli broj s predznakom, raspon brojeva od -2,147,483,648 do +2,147,483,647.
unsigned int	4	**Cjelobrojni tip bez predznaka:** 32-bitni cijeli broj bez predznaka od 0 do 4,294,967,295.
enum	4	**Pobrojavanje:** Alternativni način za izražavanje cijelog brojs s predznakom gdje neke od cjelobrojnih vrijednosti označavaju određena konstantne vrijednosti.
bool	4	**Boleova varijabla:** Logička reprezentacija cjelobrojne vrijednosti, odgovara dvorazinskom pobrojavanju pri čemu je vrijednost 0 definirana kao "FALSE" (laž) a 1 kao "TRUE" (istina)
hyper	8	**Cjelobrojni hyper tip s predznakom:** Isto kao običan cijeli broj s predznakom ali 8 bitne širine za podršku većih brojeva
unsigned hyper	8	**Cjelobrojni hyper tip bez predznaka:** Isto kao običan cijeli broj bez predznaka ali 8 bitne širine za podršku većih brojeva
float	4	**Broj sa pomičnim zarezom:** 32-bitni realni broj s predznakom. 1 bit sadrži predznak (pozitivan ili negativan), 8 bitova služi za pohranu eksponenta, a preostala 23 bita za pohranu mantise
double	8	**Broj sa pomičnim zarezom dvostruke preciznosti:** Isto kao prethodni tip ali sa više bitova za vveću preciznost. 1 bit za predznak, 11 za eksponent te 52 bita za mantisu

quadruple	16	**Broj sa pomičnim zarezom četverostruke preciznosti:** Isto kao prethodna dva tipa ali sa još više bita za veći preciznost. 1 bit za predznak, 15 za eksponent te 112 bitova za mantisu
opaque	Promjenjivo	**Nedefinirani podaci:** Podaci koji se prosljeđuju među uređajima za koje nisu dani XDR prikazi. Izraz opaque znači da se prema podacima odnosi kao prema crnoj kutiji čija je unutrašnjost nepoznata. Svaki uređaj koji koristi ovaj ti podatka očito mora znati kako izači s njim na kraj, pošto NFS ne zna.
string	Promjenjivo	**Niz:** Niz ASCII znakova proizvoljne duljine
(array)	Promjenjivo	**Polje:** Skup istovrsnih podataka bilo kojeg gore navedenog tipa može se smjestiti u polje da se omogući mnoštvu podataka da se referencira kao cjelina
struct	Promjenjivo	**Struktura:** Proizvoljna struktura koja sadrži druge podatkovne elemente iz ove tablice. To omogućuje definiciju složenih tipova podataka
void	0	**Void:** Nul tip podatka koji ne sadrži ništa
const	0	**Konstanta:** Ne deklarira podatke, samo definira konstantne vrijednosti.

Korištenje RPC protokola u klijent/poslužitelj operacijama

Gotovo sve aplikacije koriste datoteke i ostale resurse. Kada program na određenom računalu želi čitati iz datoteke, u nju pisati ili izvršiti slične radnje nad njom, mora koristiti odgovarajuće programske instrukcije. Ne bi bilo efikasno zahtijevati da svaki program sadrži kopiju tih instrukcija, pa se one ostvaruju kao standardizirani programski moduli, koji se ponekad nazivaju procedurama. Da bi izvršio radnju, dio programa poziva proceduru; procedura momentalno preuzima kontrolu i izvršava zadatak poput čitanja ili pisanja. Nakon toga procedura vraća kontrolu nazad programu koji ju je pozvao, a ako je potrebno, vraća i podatke.

Ideja NFS-a je učiniti da udaljeni pristup datotekama izgleda kao lokalni, pa se i njegov dizajn temelji na mrežnoj verziji metode pozivanja procedure upravo opisane. Aplikacija koja nešto želi učiniti sa datotekom vrši poziv procedure, ali vrši taj poziv na udaljenom računalu umjesto na lokalnom. Poseban skup rutina koristi se za rukovanje transmisijom poziva kroz mrežu, na način koji je za program koji poziv vrši nevidljiv.

Takva funkcionalnost mogla se implementirati i u sam NFS, ali je Sun umjesto toga stvorio zasebnu komponentu – *Remote Procedure Call* koja obavlja taj posao, a nalazi se u sesijskom sloju. RCP je izvorno stvoren kao potkomponenta NFS-a, ali ga, zahvaljujući korisnosti i generičnosti, koriste i druge klijentsko-serverske aplikacije u TCP/IP-u. Zbog toga se i smatra zasebnim protokolom.

Jer je RPC pravi proces komunikacije u NFS-u, NFS je sam po sebi različit od drugih TCP/IP protokola. Njegovo djelovanje ne može se opisati u uvjetima specifičnih razmjena poruka i dijagrama stanja kao što je to moguće sa protokolima poput HTTP ili DHCP jer RPC vrši sve to. NFS je u stvari definiran skupom RPC poslužiteljskih procedura i operacija koje NFS poslužitelj pruža NFS klijentima. Te procedure i operacije dopuštaju određeni tip akcije nad datotekom, poput

čitanja iz nje, pisanja u nju, ili njenog brisanja.

RCP osnove i upotreba transportnog protokola

Kada korisnik želi nešto učiniti sa datotekom na određenom uređaju, koristi RPC za stvaranje poziva NFS poslužitelju na tom uređaju. Poslužitelj prihvaća zahtjev i izvršava zahtijevanu akciju, a potom vraća rezultatni kod a možda i podatke klijentu, ovisno o zahtjevu. Rezultatni kod pokazuje uspješnost akcije. Ako je zahtjev uspješno izvršenm klijent može pretpostaviti da je, kakav god bio, zadatak izvršen. Na primjer, u slučaju pisanja podataka, klijent može pretpostaviti da su podaci uspješno zapisani u dugotrajnu memoriju.

NFS može raditi preko bilo kojeg transportnog mehanizma koji ima valjanu RPC implementaciju na sesijskom sloju. Naravno, u TCP/IP-u imamo dva transportna protokola, UDP i TCP. NFSv2 standard koristi UDP, što je još uvijek korišten način rada. NFSv3 dopušta upotrebu i UDP i TCP protokola, dok NFSv4 specificira TCP za prijenos podataka. NFS koristi 2049 broj vrata, ali u upotrebi su i druga vrata, što omogućuje RPC dio za mapiranje vrata ("port mapper").

Odgovornosti klijenta i poslužitelja

Korištenje nepouzdanog UDP protokola može se učiniti neobičnim. Na primjer, očito ne želimo da dođe do gubitka podataka koje namjeravamo upisati u datoteku. Treba međutim naglastiti da nas UDP ne spriječava da koristimo određene mjere za osiguravanje pouzdane komunikacije, on samo te mogućnosti ne podržava sam po sebi. NFS može koristiti UDP jer je protokol osmišljen da dopušta gubitak prenesenih podataka te da to popravi.

U skladu s ovim konceptom, opća ideja NFS-a stavlja većinu odgovornosti za implementaciju protokola na stranu klijenta, a ne poslužitelja. Kako u NFSv3 standardu stoji: "NFS poslužitelji su glupi a NFS klijenti su pametni". To znači da se poslužitelj usredotočuje samo na odgovaranje na zahtjeve, dok se klijentska strana mora pobrinuti za većinu zamršenih detalja protokola, uključujući oporavka od propale komunikacije. Zapravo to je vrlo čest zahtjev pri uporabi UDP-a, jer ako je zahtijev klijenta izgubljen u prijenosu, poslužitelj ni na koji način ne može znati da je taj zahtijev uopće bio poslan.

Kao što je spomenuto u pregledu NFS-a, NFS poslužitelji su "*stateless*". Jednostavnije rečeno, poslužitelj ne vodi računa o stanjima klijenata između dva zahtijeva. Svaki je zahtijev nezavisan od prethodnog, i poslužitelj u biti nema "pamćenje" o tome što je učinio prije trenutne naredbe klijenta. To opet zahtijeva dodatnu "inteligenciju" klijenta, ali istovremeno ima važnu prednost pojednostavljenja oporavka u slučaju pada poslužitelja. Pošto poslužitelj nije vodio nikakve zapise o klijentu, ništa ne može biti izgubljeno. To je važno za osiguravanje neoštećenosti datoteka pri mrežnim problemima ili zagušenjima.

Klijent/server *caching*

I NFS klijenti, a i poslužitelji mogu poboljšati performanse *caching*-om. Poslužitelji *caching* mogu koristiti za pohranu nedavno zahtijevanih informacija u slučaju da su ponovno potrebne. Također mogu koristiti prediktivni *caching* (prefetching). Tom tehnikom, poslužitelj koji prima zahtjev za čitanje bloka podataka iz datoteke može u memoriju učitati naredni blok, na temelju teoretske vjerojatnosti da će taj blok biti sljedeći zahtijevan. Sa klijentske strane *caching* se koristi da bi zadovoljilo učestale NFS zahtjeve aplikacija izbjegavajući dodatne RPC pozive. Poput svega ostalog vezanog uz NFS, *caching* je mnogo bolje podržan u NFSv4 negoli u prethodnim verzijama.

Poslužiteljske procedure i operacije

Stvarna razmjena operacija između NFS klijenta i poslužitelja odavlja se RPC protokolom. Stoga je funkcionalnost NFS-a opisana ne u uvjetima specifičnih operacija protokola, već razdvajanjem različitih akcija koje klijent može poduzeti na datotekama koje se nalaze na poslužitelju. U originalnoj inačici, NFSv2, ovi pozivi zovu se NFS poslužiteljske procedure.

Svaka procedura predstavlja određenu akciju koju klijent može izvršiti, poput čitanja iz datoteke, pisanja u nju ili stvaranja odnosno uklanjanja direktorija. Operacije izvršene nad datotekom zahtijevaju da datoteka bude referencirana korištenjem *file handle* podatkovne strukture. Kao što samo ime sugerira, ta struktura, poput drške realnog objekta, omogućuje klijentu i poslužitelju da se "zakače" za datoteku. Mount protokol koristi se za montiranje datotečnog sustava, da omogući pristup *file handle*-u NFS procedurama. Inačica 3 NFS protokola koristi isti osnovni model poslužiteljskih procedura, ali unosi određene promjene. Dvije od NFSv2 procedura su uklonjene, a nekoliko novih je dodano kao podrška novoj funkcionalnosti. Brojevi pridruženi kao identifikatori pojedinih procedura također su izmijenjeni.

NFSv2 i NFSv3 poslužiteljske procedure

Tablica 2 daje prikaz poslužiteljskih procedura definiranih u inačicama 2 i 3 NFS-a. Dani su brojevi procedura za obje inačice, kao i ime svake procedure te njen opis.

Tablica 2: NFSv2 i NFSv3 poslužiteljske procedure

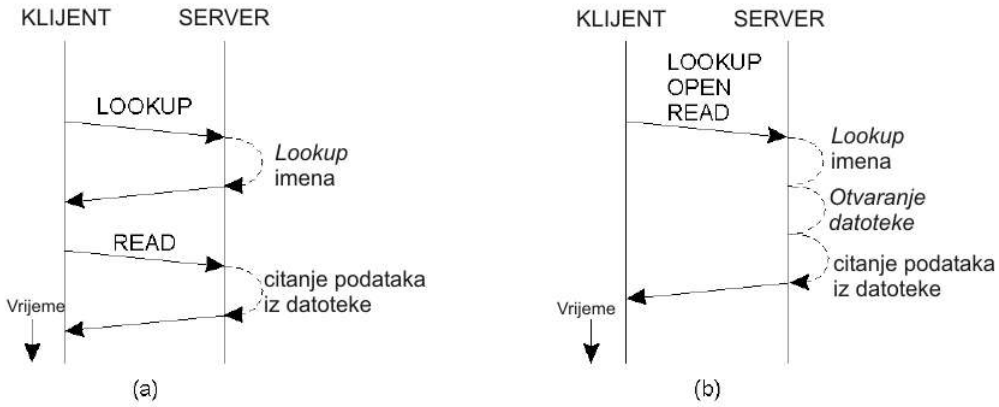
Broj procedure (v2)	Broj procedure (v3)	Naziv procedure	Sažetak	Opis
0	**0**	*null*	Ne čini ništa	"glupa" procedure namijenjena testiranju
1	**1**	*getattr*	Dohvat atributa datoteke	Dohvaća atribute datoteke na udaljenom poslužitelju
2	**2**	*setattr*	Postav atributa datoteke	Mijenja atribute datoteke na udaljenom poslužitelju
3	—	*root*	Dohvat Root-a datotečog sustava	Ova je procedura izvorno osmišljena da dopusti klijentu da nađe root udaljenog datotečnog sustava, ali se više ne koristi. Ta je funkcija sada implementirana u Mount protokol
4	**3**	*lookup*	Traženje imena datoteke	Vraća file handle datoteke klijentu

5	**5**	*readlink*	Čita iz simboličkog linka	Čita ime datoteke određene simboličkom vezom
6	**6**	*read*	Čitanje iz datoteke	Čita podatke iz datoteke
7	—	*writecache*	Pisanje u cache memorije	Proposed for future use in version 2 but abandoned and removed from version 3.
8	**7**	*write*	Pisanjeu datoteku	Zapisuje podatke u datoteku
9	**8**	*create*	Stvaranje datoteke	Stvara datoteku na poslužitelju
10	**12**	*remove*	Uklanjanje datoteke	Briše datoteku sa poslužitelja
11	**14**	*rename*	Promjena imena	Mijenja ime datoteke
12	**15**	*link*	Stvaranje veze na datoteku	Stvara “tvrdu” (ne-simboličku) vezu na datoteku
13	**10**	*symlink*	Stvaranje simbolične veze	Stvara simboličnu vezu na datoteku
14	**9**	*mkdir*	Stvaranje direktorija	Stvara direktorij na poslužitelju
15	**13**	*rmdir*	Uklanjanje direktorija	Brisanje direktorija sa poslužitelja
16	**16**	*readdir*	Čitanje iz direktorija	Čita sadržaj direktorija
17	—	*statfs*	Dohvat atributa datotečnog sustava	Pruža klijentu opće informacije o udaljenom datotečnom sustavu, uključujući njegovu veličinu i količinu preostalog slobodnog prostora U NFSv3 zamijenjeno sa *fsstat* i fsinfo
—	**4**	*access*	Provjera dozvole pristupa	(Novo u NFSv3.) Određuje korisnička prava pristupa za određene objekte datotečnog sustava.
—	**11**	*mknod*	Stvaranje posebnog uređaja	(Novo u NFSv3.) Stvaranje posebne datoteke poput cjevovoda ili uređaja

—	**17**	*readdirplus*	Prošireno čitanje iz direktorija	(Novo u NFSv3.) Dohvat dodatnih informacija iz direktorija
—	**18**	*fsstat*	Dohvat dinamičkih informacija datotečnog sustava	(Novo u NFSv3.) Vraća promjenjive informacije o stanju datotečnog sustava poput trenutne količine slobodnog prostora datotečnog sustava
—	**19**	*fsinfo*	Dohvat statičnih informacija datotečnog sustava	(Novo u NFSv3.) Vraća statičke informacije o datotečnom sustavu npr. O načinu korištenja datotečnog sustava, te parametrima kako zahtijevi poslužitelju trebaju biti strukturirani
—	**20**	*pathconf*	Dohvat POSIX informacija	(Novo u NFSv3.) Dohvaća dodatne informacije za datoteku ili direktorij

Često se dešava da klijent želi izvršiti višestruke akcije nad datotekom: nekoliko uzastopnih čitanja, primjerice. Jedan od problema sa sustavom poslužiteljskih procedura u NFSv2 i NFSv3 jest da svaka klijentska akcija zahtijeva zaseban poziv procedure.

U svrhu poboljšanja poslužiteljskih procedura, NFSv4 uvodi značajne promjene u način na koji su poslužiteljske procedure implementirane. Umjesto da je svaka klijentska akcija zasebna procedura, definira se novi tip – *compound* procedura. Unutar takve, compound procedure, učahuren je velik broj poslužiteljskih operacija. One se sve šalju kao jedinstvena jedinica te poslužitelj interpretira i prati instrukcije svake operacije u nizu.



Ilustracija 2(a) čitanje iz datoteke u NFSv3, (b) čitanje podataka korištenjem compound procedure u NFSv4

Ta promjena znači da postoje samo dvije procedure u NFSv4, kao što je to prikazano u narednoj tablici.

Tablica 3: NFSv4 posluiteljske procedure

Broj procedure	ime procedure	Sažetak	Opis
0	*null*	Ne čini ništa	“glupa” procedura namijenjena testiranju
1	*compound*	Compound operacije	Kombinira više NFS operacije u jedinstven zahtjev, kao što je opisano

Sve su stvarne klijentske akcije definirane kao operacije unutar coumpound procedure, kako je u tablici 4 prikazano. Vidi se da su brojevi NFSv4 operacija mnogo veći od brojeva procedura u NFSv2 i NFSv3. Do toga je došlo uslijed dodanih mogućnosti u inačici 4 i činjenice da NFSv4 funkcije odvojenog Mount protokola integrira u sam NFS.

Tablica 4: NFSv4 poslužiteljske operacije

Broj operacije	Broj operacije	Sažetak	Opis
3	*access*	Provjera prava pristupa	Determines the access rights a user has for an object.
4	*close*	Zatvaranje datoteke	Closes a file.
5	*commit*	Izvršavanje podataka u cache memoriji	Uklanja sve podatke u poslužiteljskoj zapisnoj cache memoriji, da bi se osiguralo da su podaci na čekanju trajno spremljeni
6	*create*	Stvaranje neregularnog datotečnog objekta	Ovo je slično mknod procedure u NFSv3; stvara “neregularni” (posebni) datotečni objekt (Regularne datoteke stvaraju se open operacije)
7	*delepurge*	Otkazivanje delegacija koje čekaju oporavak	NFSv4 ima mogućnost da poslužitelj može delegirati klijentu odgovornost za određene datoteke. Ova operacija uklanja delegacije koje čekaju oporavak
8	*delegreturn*	Vraćanje delegacija	Vraća delegacije klijenta poslužitelju koji ju je odobrio
9	*getattr*	Dohvat atributa	Dohvat atributa datoteke
10	*getfh*	Dohvat trenutnog *file handle*-a	Vraća *file handle*, logički objekt koji se koristi da bi se dopustio pristup datoteci
11	*link*	Stvaranje veze na datoteku	Stvaranje “tvrde” (ne-simboličke) veze na datoteku

12	*lock*	Stvaranje “lokota”	Stvaranje “ključanice” da datoteci. “ključanica” se koristi za rukovanje pristupom datoteci, npr. Sprećavanje dva klijenta da naizmjence pišu u datoteku i tako ju blokiraju
13	*lockt*	Test za “lokot”	Ispitivanje postojanja “lokota” na objektu i vraćanje informacije o tome
14	*locku*	Otključavanje datoteke	Uklanja prethodno stvoren “lokot” na datoteci
15	*lookup*	Traženje imena datoteke	Traženje datoteke.
16	*lookupp*	Roditeljski direktorij	Vraća *file handle* roditeljskog direktorija objekta
17	*nverify*	Provjera razlike u atributima	Provjerava da li su se atributi datoteke promijenili
18	*open*	Otvaranje regularne datoteke	Otvara datoteku
19	*openattr*	Otvaranje imenovanog direktorija atributa	Otvara direktorij atributa vezan uz datoteku
20	*open_confirm*	Potvrda otvaranja	Potvrđivanje informacije vezane uz otvaranje datoteke
21	*open_ downgrade*	Smanjenje pristupa otvorenoj datoteci	Postavlja prava pristupa za datoteku koja je već otvorena
22	*putfh*	Postav trenutnog *Filehandle*-a	Zamjenjuje jedan *filehandle* s drugim.
23	*putpubfh*	Postavljanje javnog *Filehandle*-a	Postavlja trenutni filehandle kao “javni” *filehandle* poslužitelja. Ovo može ali i ne more biti isto kao root *filehandle* (dolje).
24	*putrootfh*	Postavljanje Root *Filehandle*-a	Postavlja trenutni *filehandle*-a kao root-a poslužiteljskog datotečnog sustava
25	*read*	Čitanje iz datoteke	Čita podatke iz datoteke
26	*readdir*	Čitanje direktorija	Čita sadržaj direktorija

27	*readlink*	Čitanje simboličke veze	Čita ime specificirane datoteke korištenjem simboličke veze
28	*remove*	Uklanjanje objekta datotečnog sustava	Uklanjanje (brisanje) objekta
29	*rename*	Preimenovanje zapisa direktorija	Mijenja ime objekta
30	*renew*	Obnavljanje zahtijeva	Obnavlja NFS delegaciju koju je stvorio poslužitelj
31	*restorefh*	Obnavljanje spremljenog *filehandle*-a	Omogućuje da se prethodno spremljeni *filehandle* učini trenutnim
32	*savefh*	Spremanje trenutnog *filehandle*-a	Omogućuje spremanje filehandle-a za kasniju upotrebu
33	*secinfo*	Dobavljanje dostupne sigurnosti	Dohvaća informacije o NFS sigurnosti
34	*setattr*	Postavljanje atributa	Mijenja jedan ili više atributa datoteke
35	*setclientid*	Pregovaranje klijenta	Dopušta klijentu komunikaciju sa poslužiteljem za prosljeđivanje informacije o načinu korištenja NFS-a
36	*setclientid_ confirm*	Potvrda *klijentid*-a	Koristi se za potvrdu rezultata prethodnog pregovaranje korištenjem setclientid
37	*verify*	Potvrda atributa	Dopušta klijentu da potvrdi određene atribute prije nastavka određene akcije
38	*write*	Pisanjeu datoteku	Zapisuje podatke u datoteku
10044	*illegal*	Ilegalna operacija	Operacija koja se koristi za podršku izvještaja o greškama kada se koriste nevažeće operacije u zahtjevima klijenta

NFS model datotečnog sustava i mount protokol

Kako klijent koristi NFS za simulaciju pristupa udaljenim direktorijima kao da su lokalni, protokol mora “predstaviti” datoteke udaljenog sustava lokalnom korisniku. Baš kao što su datoteke na lokalnom uređaju za pohranu uređene korištenjem određenog datotečnog sustava, NFS koristi model datotečnog sustava za predstavljanje prikaza datoteka korisniku.

Model datotečnog sustava

Model datotečnog sustava što ga koristi NFS je isti onaj sa kojim smo svi upoznati: hijerarhijski raspored direktorija koji sadrže datoteke i poddirektorije. Na vrhu hijerarhije nalazi se root, koji sadrži bilo koji broj datoteka i diretorije prve razine. Svaki direktorij može sadržavati više datoteka ili drugh direktorija, što omogućuje proizvoljnu strukturu datotečnog stabla.

Datoteka je jednoznačno određena imenom datoteke i nazivom puta koji prikazuje niz direktorija kojima se putuje od roota do datoteke. Kako je NFS povijesno vezan uz UNIX, i datoteke su u NFS-u često prikazane UNIX notacijom.

Mount protokol

Prije nego li se NFS može koristiti za pristup datoteci na udaljenom računalu, klijentu se na određen način mora omogućiti pristup toj datoteci. To znači da dio udaljenog datotečnog sustava mora biti dostupan klijentu. Pri stvaranju NFS protokola odlučeno je da se pristup datotekama, njihovo otvaranje i zatvaranje ne ugradi u sam protokol. Umjesto toga je stvoren zaseban protokol da podrži rad NFS-a, tako da ako u budućnosti bude potrebno mijenjati metode pružanja pristupa datotekama, to ne zahtijeva promjene u samom NFS-u. Spomenuti odvojeni mehanizam zove se Mount protokol i opisan je u odjeljku A RFC-a 1094 (NFSv2). Valja primijetiti da iako je fuknkcionalno zaseban, Mount se smatra dijelom ukupnog NFS paketa.

Pri nadogradnji na treću inačicu NFS protokola, Mount protokol je blago izmijenjen. Za tu inačicu NFS-a, Mount protokol je definiran u Odjeljku I RFC 1813 (NFSv3). Sadrži neke promjene u samom radu protokola, ali je opće djelovanje uglavnom sačuvano istim.

Izraz “montiranja” je zapravo analog hardverskog izraza koji se odnosi na postupke da bi se fizičke jedinice za pohranu učinile dostupnima. U prijašnja vremena, uređaji za pohranu su najčešće bili uklanjajući diskovi, i da bi se koristili, bilo ih je potrebno montirati na jedinicu uređaja. Na sličan su način logički montirani NFS resursi korištenjem Mount protokola, što omogućuje dostupnost dijeljenog datotečnog sustava klijentu. Datoteka može biti otvorena i *file handle* vraćen klijentu tako da pokazuje na datoteku sa kojom se nešto želi raditi.

Poslužiteljske procedure Mount protokola

Implementacija Mount protokola vrlo je slična onoj samog NFS protokola, Poput NFS-a, i Mount protokol koristi XDR za opis tipova podataka koji se razmijenjuju između klijenta i poslužitelja, te RPC za definiranje skupa poslužiteljskih procedura koje su klijentu na raspolaganju za obavljanje različitih operacija. Glavna je razlika Mount i NFS protokola jednostavno u tome što Mount protokol definira procedure vezane uz otvaranje i zatvaranje datotečnog sustava, a ne operacije pristupa datoteci. Tablica 4. prikazuje poslužiteljske procedure korištene u Mount protokolu.

Tablica 5: NFS Poslužiteljske procedure Mount protokola

Broj procedure	Ime procedure	Sažetak	Opis
0	*null*	Ne čini ništa	“glupa” procedura za testiranje
1	*mnt*	Dodaj Mount zapis	Vrši operaciju montiranja mapiranjem puta sa poslužitelja na *file handle*

2	*dump*	Vrati Mount zapise	Vraća listu udaljeno montiranih datotečnih sustava
3	*umnt*	Ukloni Mount zapis	Vrši “demontiranje” uklanjanjem mount zapisa
4	*umntall*	Uklanjanje svih Mount zapisa	Uklanja sve mount zapise, eliminirajući sve montirane datotečne sustave između poslužitelja i klijenta
5	*export*	Vraćanje izvozne liste	Vraća listu izvezenih datotečnih sustava i pokazuje koji klijenti imaju pravo njihovog montiranja. To se koristi da bi klijent vidio koji su datotečni sustavi na raspolaganju za upotebu.