VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV INFORMAČNÍCH SYSTÉMŮ

FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF INFORMATION SYSTEMS

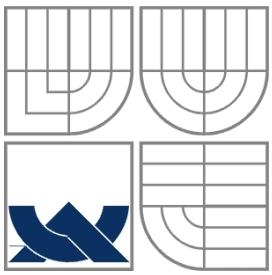
NNTP SERVER JAKO SLUŽBA PRO SYSTÉMY ZALOŽENÉ NA TECHNOLOGII WINDOWS-NT

DIPLOMOVÁ PRÁCE
MASTER‘S THESIS

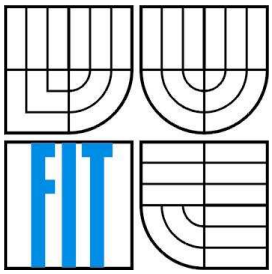
AUTOR PRÁCE
AUTHOR

BC. JOSEF LOUPANEC

BRNO 2007



VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY



FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV INFORMAČNÍCH SYSTÉMŮ

FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF INFORMATION SYSTEMS

NNTP SERVER JAKO SLUŽBA PRO SYSTÉMY ZALOŽENÉ NA TECHNOLOGII WINDOWS-NT

NNTP SERVER AS A WINDOWS NETWORK SERVICE

DIPLOMOVÁ PRÁCE
MASTER'S THESIS

AUTOR PRÁCE — BC. JOSEF LOUPANEC
AUTHOR

VEDOUCÍ PRÁCE — ING. PAVEL OČENÁŠEK
SUPERVISOR

BRNO 2007

NNTP server jako služba pro systémy založené na technologii Windows-NT

Vedoucí:

Očenášek Pavel, Ing., UIFS FIT VUT

Přihlášen:

Loupanec Josef, Bc.

Zadání:

1. Seznamte se s protokoly NNTP, HTTP, případně s jejich zabezpečenými variantami, podle příslušných RFC.
2. Seznamte se s možnostmi tvorby aplikací v prostředí Windows-NT (resp. Win-XP). Především se zaměřte na implementaci služeb (services).
3. Navrhněte vlastní diskuzní server umožňující práci s uvedenými protokoly. Server bude umět pracovat s lokálními diskuzními složkami (víceuživatelský přístup) a případně předávat požadavky na vzdálené diskuzní servery.
4. Do návrhu zahrňte vzdálené stahování diskuzních příspěvků z jiných serverů.
5. Navržený server implemetujte jako službu pro prostředí Win-NT.
6. Zhodnoťte dosažené výsledky a diskutujte další možnosti rozšíření.

Část požadovaná pro obhajobu SP:

Body 1 - 4.

Kategorie:

Počítačové sítě

Operační systém:

Windows-NT (Win2000, WinXP, ...)

Literatura:

- Příslušná RFC k jednotlivým protokolů s formátům zpráv.
- Dále dle doporučení vedoucího práce.

Komentář:

Server musí podporovat autentizaci uživatele.

LICENČNÍ SMLOUVA
POSKYTOVANÁ K VÝKONU PRÁVA UŽÍT ŠKOLNÍ DÍLO

uzavřená mezi smluvními stranami

1. Pan

Jméno a příjmení: Bc. Josef Loupanec
Id studenta: 49263
Bytem: Šléglov 4, 78825
Narozen: 21.9.1982 v Šumperku
(dále jen "autor")

a

2. Vysoké učení technické v Brně

Fakulta informačních technologií
se sídlem Božetěchova 2/1, 612 66 Brno, IČO 00216305
jejímž jménem jedná na základě písemného pověření děkanem fakulty:

.....
(dále jen "nabyvatel")

Článek 1
Specifikace školního díla

1. Předmětem této smlouvy je vysokoškolská kvalifikační práce (VŠKP):
diplomová práce

Název VŠKP: NNTP Server jako služba pro systémy založené na technologii Windows-NT
Vedoucí/školitel VŠKP: Ing. Pavel Očenášek
Ústav: Ústav informačních systémů
Datum obhajoby VŠKP: .....

VŠKP odevzdal autor nabyvateli v:

tištěné formě počet exemplářů: 1
elektronické formě počet exemplářů: 2 (1 ve skladu dokumentů, 1 na CD)

2. Autor prohlašuje, že vytvořil samostatnou vlastní tvůrčí činností dílo shora popsané a specifikované. Autor dále prohlašuje, že při zpracovávání díla se sám nedostal do rozporu s autorským zákonem a předpisy souvisejícími a že je dílo dílem původním.
3. Dílo je chráněno jako dílo dle autorského zákona v platném znění.
4. Autor potvrzuje, že listinná a elektronická verze díla je identická.

Článek 2
Udělení licenčního oprávnění

1. Autor touto smlouvou poskytuje nabyvateli oprávnění (licenci) k výkonu práva uvedené dílo nevýdělečně užít, archivovat a zpřístupnit ke studijním, výukovým a výzkumným účelům včetně pořizování výpisů, opisů a rozmnoženin.
2. Licence je poskytována celosvětově, pro celou dobu trvání autorských a majetkových práv k dílu.
3. Autor souhlasí se zveřejněním díla v databázi přístupné v mezinárodní síti:
   - ☐ ihned po uzavření této smlouvy
   - ☐ 1 rok po uzavření této smlouvy
   - ☒ 3 roky po uzavření této smlouvy
   - ☐ 5 let po uzavření této smlouvy
   - ☐ 10 let po uzavření této smlouvy

   (z důvodu utajení v něm obsažených informací)
4. Nevýdělečné zveřejňování díla nabyvatelem v souladu s ustanovením § 47b zákona č. 111/ 1998 Sb., v platném znění, nevyžaduje licenci a nabyvatel je k němu povinen a oprávněn ze zákona.

Článek 3
Závěrečná ustanovení

1. Smlouva je sepsána ve třech vyhotoveních s platností originálu, přičemž po jednom vyhotovení obdrží autor a nabyvatel, další vyhotovení je vloženo do VŠKP.
2. Vztahy mezi smluvními stranami vzniklé a neupravené touto smlouvou se řídí autorským zákonem, občanským zákoníkem, vysokoškolským zákonem, zákonem o archivnictví, v platném znění a popř. dalšími právními předpisy.
3. Licenční smlouva byla uzavřena na základě svobodné a pravé vůle smluvních stran, s plným porozuměním jejímu textu i důsledkům, nikoliv v tísni a za nápadně nevýhodných podmínek.
4. Licenční smlouva nabývá platnosti a účinnosti dnem jejího podpisu oběma smluvními stranami.

V Brně dne: .....

..... Josef Lompanec

Nabyvatel Autor

Abstrakt:

Tato práce se zabývá analýzou požadavků, návrhem a implementací internetového diskusního serveru. Přesněji řečeno, jedná se o server spravující diskusní skupiny s příspěvky a zajišťující jejich dostupnost prostřednictvím protokolu NNTP a HTTP. Server podporuje autentizaci uživatele a disponuje volitelným proxy módem, kdy jsou všechny NNTP požadavky přeposílány na vzdálený NNTP diskusní server. Součástí programu je též mechanismus zajišťující stahování příspěvků ze vzdálených NNTP serverů a tím pádem plnící distribuční funkci. Aplikace je určena pro operační systémy MS Windows počínaje verzí NT a vyšší. Program poběží jako služba NT a je konfigurovatelný prostřednictvím grafického uživatelského rozhranní. V dokumentu jsou také obsaženy teoretické informace nutné pro praktické zvládnutí výše uvedených kroků.

Klíčová slova:

Relační databáze, HTTP protokol, NNTP protokol, proxy server, regulární výraz, formát diskusního příspěvku, služba Windows, SQL, MySQL, server, diskusní skupina, distribuce, HTML, port, URL, vlákno, service control manager, relace, primární klíč,

Abstract:

This work includes specification and analysis of requirements, design and implementation of the internet news server. The server controls newsgroups and associated news. It provides availability of the articles by NNTP protocol and HTTP protocol (by web interface). The server supports a user authentication and an optional proxy mode, when all NNTP requests are resent to another remote NNTP server. A mechanism that provides news-downloading from remote NNTP servers and performs distribution function is included too. The application is designed to run on MS Windows NT (and higher version) as a NT service. The server is configurable by a graphic user interface. The work also includes theoretical information needed for successful accomplishment of the above-mentioned requirements.

Key Words:

Relation Database, HTTP, NNTP, proxy server, regular expression, news format, Windows service, SQL, MySQL server, newsgroup, distribution, HTML, port, URI, thread, service control manager, relation, primary key

Josef Loupanec: NNTP server jako služba pro systémy založené na technologii Windows NT, **diplomová práce, Brno, FIT VUT v Brně,** 2007

Prohlášení

Prohlašuji, že jsem tuto diplomovou práci vypracoval samostatně pod vedením Ing. Pavla Očenáška. Uvedl jsem všechny literární prameny a publikace, ze kterých jsem čerpal.

Obsah

Úvod

Cílem tohoto projektu je analýza požadavků, návrh a dále implementace diskusního NNTP serveru určeného pro platformu Microsoft Windows (počínaje verzí NT). Podstatnou částí tohoto projektu je též zahrnutí teoretických znalostí nutných pro úspěšné zvládnutí výše uvedených kroků.

První část dokumentu obsahuje upřesnění specifikace požadavků na systém a dále je zaměřena především na teoretičtější informace nutné pro vytvoření projektu. Nejprve se jedná o popisy protokolu HTTP a NNTP potřebných pro implementaci internetových rozhraní serveru. Dále je zde popsán princip práce s relační databází a její použití pro uložení diskusních příspěvků. Součástí bude také vysvětlení principu fungování a ovládání služeb systémů Windows. Zmíněny jsou regulární výrazy a jejich využití při lexikální analýze. Druhá část dokumentu se zabývá návrhem jednotlivých částí diskusního serveru. Jmenovitě se jedná o NNTP a HHTP rozhraní, databázové schéma, klienta zabezpečujícího distribuci zpráv a uživatelské rozhraní. Třetí část práce se věnuje popisu implementace navržených komponent a způsobu splnění kritérií kladených na služby systému Windows. Manuál k vytvořené aplikaci je obsažen v přílohách této práce.

Tato diplomová práce navazuje na předešlý semestrální projekt [8], ze kterého jsou převzaty některé teoretické informace v úvodu práce. Teorie týkající se NNTP protokolu je však vytvořená nově, neboť v době práce na projektu bylo vydáno RFC 3977 definující novou verzi protokol NNTP, kterou se tento projekt řídí. Z tohoto důvodu jsou též přepracovány kapitoly týkající se návrhu projektu, jejichž nosná část je však inspirována také semestrálním projektem.

Upřesnění požadavků

Na tomto místě jsou upřesněny ty požadavky, které nebyly přesně specifikované v rámci zadání a jejichž volba byla ponechána na autorovi projektu. Pro implementaci bude použit programovací jazyk Java s podporou rozhranní JDBC pro prácí s databází. Pro daný projekt byl zvolen databázový systém MySQL, především díky tomu, že je pro nekomerční aplikace k dispozici zdarma. Vzhledem k tomu, že implementace grafického uživatelského rozhraní pro služby systému Windows je problematická, bude toto rozhranní implementováno jako samostatná aplikace, která pouze edituje konfigurační soubor, který si načítá služba při svém spuštění, resp. restartu. Pomocí této konfigurační aplikace bude též možné provádět operace s databází (vytváření/rušení diskusních skupin, uživatelů, ...)

Teoretické informace

NNTP protokol

Úvod

Původní Network News Transfer Protocol (NNTP) je popsán v RFC 977. Jeho vznik je datován do února roku 1986. V říjnu roku 2006 vychází nové RFC 3977 definující tento protokol, které přináší mnoho změn a rozšíření oproti původnímu RFC 977.

NNTP specifikuje protokol pro distribuci, nalezení, přenos a zasílání diskusních příspěvků pomocí protokolu TCP/IP v rámci sítě Internet. Je navržen tak, že články jsou uloženy v centrální databázi a uživatel si může vybrat a stáhnout pouze ty články, které chce číst. Tento protokol též poskytuje indexování, křížové odkazy a exspiraci neaktuálních zpráv.

Hlavním důvodem pro zavedení protokolu NNTP je obrovské rozšíření používání diskusních zpráv mezi uživateli Internetu. V dnešní době tento protokol již téměř úplně vytlačil diskusní skupiny založené na seznamech emailových adres, neboť jejich používání se ukázalo jako neefektivní hlavně z hlediska plýtvání přenosový pásmem a nutností přijímat i nevyžádané zprávy. Tyto nedostatky jsou u NNTP eliminovány právě existencí centrálního diskusního serveru vybaveného centrální databází s diskusními příspěvky, ze kterých si může uživatel stáhnout a přečíst pouze ty příspěvky, které jsou předmětem jeho zájmu.

Důležitou vlastností protokolu NNTP je podpora distribuce diskusních zpráv. Součástí jsou totiž příkazy, které poskytují přímou metodu výměny článků mezi spolupracujícími počítačovými systémy. Distribuce u NNTP nahrazuje zastaralý způsob distribuce zpráv pomocí zaplavování, kdy každý systém zasílá všechny zprávy svým sousedům v distribuční skupině. Tento způsob distribuce však vede k neúměrnému plýtvání zdroji jak v časové tak datové oblasti. Navíc přináší problém redundance a následné nutnosti detekce duplikátů a jejich mazání.

Výměna článků při použití protokolu NNTP probíhá následujícím způsobem. Systémy provádějící výměnu diskusních zpráv mají interaktivní mechanismus pro rozhodování, které články se mají přenášet. Systém, který vyžaduje nové zprávy, nebo který má zprávy určené pro zaslání, typicky kontaktuje jednoho nebo více svých sousedů pomocí protokolu NNTP. Nejprve pomocí příkazu NEWGROUPS zjistí, jestli byly na kontaktovaném systému vytvořeny nějaké nové diskusní skupiny. Pokud se tak stalo (a na základě lokálních pravidel je dovolena distribuce těchto nových skupin) jsou tyto skupiny založeny i na lokálním systému.

Klient potom zjistí, zda do skupin, které jsou předmětem jeho zájmu byly doručeny nějaké nové články. Prostředkem k tomuto zjištění je příkaz NEWNEWS. Tento příkaz doručí seznam nových článků na serveru a na základě tohoto seznamu může začít sekvence přenosů článků ze serveru na klienta. Nakonec může klient nabídnout serveru ty nové články, které klient často dostává. Server se

potom rozhodne, které z těchto článků již obdržel, a které by měli být zaslány a přidány do jeho sbírky.

NNTP Přehled

NNTP server specifikovaný v RFC 3977 používá textové příkazy a odpovědi podobně jako např. protokol SMTP. Je navržen tak, aby akceptoval příchozí připojení od klientů a poskytoval jednoduché rozhraní k databázi s diskusními příspěvky. Jako transportní vrstva síťového protokolu je použit protokol TCP a pro tuto službu je jako výchozí vyhrazen port 119.

Příkazy a odpovědi jsou složené ze znaků v kódování UTF-8. Příkazy se skládají z příkazových slov, které jsou v některých případech následovány jedním nebo více parametry. Parametry musí být od sebe a od příkazového slova odděleny jednou nebo více mezerami a nebo tabelátorem. Příkazová řádka musí obsahovat všechny povinné parametry daného příkazu a nemůže obsahovat více než jeden příkaz. Příkazy ani parametry nejsou citlivé na velikosti písmen. Příkaz musí být ukončen dvojicí znaků CR-LF (Carriage Return – Line Feed). Délka příkazové řádky nesmí být větší jak 512 znaků (započítávají se všechny mezery, tabelátory i koncová dvojice CR-LF). Tj. pro příkaz a jeho parametry je použitelných 510 znaků. Neexistuje žádná možnost jak pokračovat příkazový řádek. Server může tolerovat překročení tohoto limitu pro příkazy definované v rozšířeních tohoto protokolu.

Z jistých důvodů nejsou všechny příkazy ve specifikaci RFC 3977 povinné. Ačkoliv některé příkazy povinné jsou a musí je implementovat každý NNTP server. Ostatní příkazy jsou rozděleny do balíků, které jsou pojmenované a to zda jsou na serveru implementovány určuje, zda je jméno balíku do kterého náležejí uvedené v seznamu schopností serveru.

NNTP Server je tradičně využíván dvěma různými způsoby. Prvním způsobem použití je doručování, kdy klient stahuje články z velkého úložiště obhospodařovaného NNTP serverem. Druhým použitím je ukládání a distribuce článků jiným serverů. V praxi jsou tyto dva způsoby použití NNTP protokolu natolik odlišné, že se vytvářejí implementace serverů specializované buď na první, nebo druhé použití, což je doporučováno i tímto RFC. Existuje však kombinace obou dvou těchto módů v jedné implementaci, takový server se nazývá *mode-switching* server.

WILDMAT formát

WILDMAT formát byl vyvinut, aby poskytoval jednotný mechanismus pro nalezení řetězce, který vyhovuje určitému vzoru, s podobnou funkčností jakou disponuje např. Unixový shell při výběru souborů. Popsán je následující ABNF syntaxí:

```
wildmat = wildmat-pattern *("," ["!"] wildmat-pattern)
     wildmat-pattern = 1*wildmat-item
     wildmat-item = wildmat-exact / wildmat-wild
```

```
wildmat-exact = %x22-29 / %x2B / %x2D-3E / %x40-5A / %x5E-7E /
     UTF8-non-ascii ; exclude ! * , ? [ \ ]
wildmat-wild = "*" / "?"
```

WILDMAT je testován oproti řetězci, kterému buď vyhovuje, nebo nevyhovuje. K provedení takového porovnání, je každý *wildmat-pattern*, ze kterého se WILDMAT skládá, porovnán s řetězcem a *wildmat-pattern* nacházející se nejvíc napravo, který se shoduje s předloženým řetězcem, je identifikován. Pokud není *wildmat-pattern* předcházen znakem „!“, shoduje se celý WILDMAT. Pokud je předcházen znakem „!“, nebo pokud se neshoduje žádný *wildmat-pattern*, celý WILDMAT se neshoduje. *Wildmat-exact* se shoduje se stejným znakem, jaký jej tvoří. Znak „?“ odpovídá právě jednomu znaku. Znak „*“ se shoduje se sekvencí znaků délky nula a více.

Příkazy administrace sezení

Založení spojení.

Tento příkaz nezasílá klient serveru, ale server musí prezentovat vhodnou odpověď jako přivítání pro klienta, který se k serveru připojil. Odpověď informuje klienta, zda je služba dostupná a zda je klientovi umožněno zasílat články na diskusní server. Zde jsou možné odpovědi:

```
200    Service available, posting allowed
201    Service available, posting prohibited
400    Service temporarily unavailable
502    Service permanently unavailable
```

Capabilities

Tento příkaz je povinný a musí být implementován každým NNTP serverem.

Syntaxe:

```
CAPABILITIES [keyword]
```

Odpověď:

```
101    Capability list follows (multi-line)
```

Tento příkaz umožňuje klientovi zjistit schopnosti serveru. Schopnosti serveru navracené během sezení mohou být změněny prostřednictvím nějakého jiného příkazu (např. MODE READER, pro přepnutí módu u *mode-switching* serveru). První řádek odpovědi na tento příkaz začíná řetězcem VERSION a pokračuje číselným určením verze protokolu NNTP, který tento server implementuje.

Mode Reader

Syntaxe:

```
MODE READER
```

Odpovědi:

```
200    Posting allowed
201    Posting prohibited
502    Reading service permanently unavailable [1]
```

Možnost použití tohoto příkazu je indikována schopností serveru MODE-READER. A slouží k přepnutí u *mode-switching* serveru z módu určeného pro distribuci příspěvků na mód doručování příspěvků z lokální databáze klientům.

Quit

Syntaxe:

```
QUIT
```

Odpovědi:

```
205    Connection closing
```

Tento příkaz používá klient k ukončení spojení.

Příkazy pro zasílání a stahování článků

Group

Syntaxe:

```
GROUP group
```

Odpovědi:

```
211 number low high group     Group successfully selected
411                           No such newsgroup
```

Příkaz je indikován schopností READER. Povinný parametr `group` je jméno diskusní skupiny, která má být vybrána. Odpověď na úspěšný příkaz *GROUP* obsahuje číslo prvního a posledního článku ve skupině a očekávaný počet článků ve skupině. Tento počet nemusí být přesný. Musí být pouze roven nebo větší než počet článků, které se ve skupině doopravdy vyskytují. Pokud je skupina úspěšně vybrána, vnitřně udržovaný ukazatel akt. článku je nastaven na první článek ve skupině. Pokud je vybrána chybná skupina, předtím vybraná skupina ani ukazatel na aktuální článek se nemění. V případě, že je vybrána prázdná skupina, je ukazatel na aktuální článek nenastaven a neměl by být používán, neboť jeho použití způsobí chybu.

Listgroup

Syntaxe:

```
LISTGROUP [group [range]]
```

Odpovědi:

```
211 number low high group     Article numbers follow (multi-line)
411                           No such newsgroup
412                           No newsgroup selected [1]
```

Přítomnost příkazu je indikována schopností READER. Příkaz může být následován nepovinným parametrem *group* (název diskusní skupiny) a tento nepovinný parametr může být následován dalším parametrem *range*. Příkaz se vnitřně chová naprosto stejně jako příkaz GROUP uvedený výše, tj. vnitřní ukazatel na aktuální skupinu je nastavován stejným způsobem. Rozdíl je však v odpovědi, kdy po řádku se stavovým kódem 211 (jeho formát je také shodný s tím u příkazu GROUP) následují řádky s čísly jednotlivých článků v dané diskusní skupině. Rozsah čísel, který se má zobrazit může být omezen právě parametrem *range*. Ten může nabývat následujících tří tvarů:

- číslo článku
- číslo článku následované znakem „-“, což určuje všechny následující čísla od zadaného čísla
- číslo1 článku, znak „-“, číslo2 článku, což určuje výpis všech čísel článků mezi číslem1 a číslem2.

Last

Syntaxe:

```
LAST
```

Odpovědi:

```
223 n message-id    Article found
412                 No newsgroup selected
420                 Current article number is invalid
422                 No previous article in this group
```

Přítomnost tohoto příkazu je indikována schopností READER. Ukazatel na aktuální článek je tímto příkazem nastaven na předchozí článek v aktuální skupině. Pokud je před tímto příkazem nastaven na první článek ve skupině, je navrácena chyba a nastavení ukazatele akt. článku se nezmění. Odpověď na úspěšné provedení tohoto příkazu obsahuje číslo aktuálního článku v dané diskusní skupině a dále Message-ID článku.

Next

Syntaxe:

```
NEXT
```

Odpovědi:

```
223 n message-id    Article found
412                 No newsgroup selected
420                 Current article number is invalid
421                 No next article in this group
```

Tento příkaz nastaví ukazatel aktuálního článku na další článek v aktuální diskusní skupině. Pokud již ve skupině nejsou žádné další články je navrácena chybová zpráva a ukazatel aktuálního článku zůstává nezměněn. Při úspěšném provedení příkazu je v odpovědi navráceno číslo aktuálního článku a jeho identifikátor.

ARTICLE, BODY, HEAD, and STAT

Syntaxe:

```
ARTICLE message-id
ARTICLE number
ARTICLE
```

Odpovědi:

První tvar (pomocí message-id):

```
220 0|n message-id    Article follows (multi-line)
430                   No article with that message-id
```

Druhý tvar (pomocí čísla článku)

```
220 n message-id      Article follows (multi-line)
412                   No newsgroup selected
423                   No article with that number
```

Třetí tvar (aktuální článek)

```
220 n message-id      Article follows (multi-line)
412                   No newsgroup selected
420                   Current article number is invalid
```

ARTICLE - Celý text článku je navrácen jako textová odpověď.
HEAD – vrací pouze řádky hlavičky
BODY – vrací pouze tělo zprávy.
STAT – nevrací žádný text. Při selekci pomocí čísla zprávy nastavuje vnitřně udržovaný ukazatel na aktuální článek.

Přítomnost těchto příkazů je indikována schopností READER. Existují dvě formy těchto příkazů. Každá používá jinou metodu k určení, který článek má být doručen. Pokud jsou tyto příkazy následovány identifikátorem zprávy v lomených závorkách. Je např. při použití příkazu ARTICLE zobrazena hlavička, prázdná řádka a potom tělo specifikovaného článku. Nedochází však k přenastavení ukazatele na aktuální článek. Pokud je za příkazem uveden nepovinný parametr číslo článku v dané diskusní skupině, je zobrazen článek s daným pořadovým číslem. Tento postup vede k nastavení ukazatele na aktuální článek. Pokud je určeno chybné číslo (např. mimo rozsah článků ve skupině) k přednastavení ukazatele aktuálního článku nedojde. V případě, že je číselný parametr vynechán je zobrazen článek, na který ukazuje ukazatel akt. článku. Není-li ukazatel akt. článku

nastaven, je signalizována chyba. Kladná odpověď vždy obsahuje číslo článku, kterého se týká, v případě, že šlo o formu příkazu s *Message-ID* je číslo článku nahrazeno nulou.

Post

Syntaxe:

```
POST
```

Odpovědi:

Úvodní odpovědi:

```
340    Send article to be posted
440    Posting not permitted
```

Následující odpovědi:

```
240    Article received OK
441    Posting failed
```

Přítomnost tohoto příkazu je indikována schopností POST. Pokud je zasílání do skupiny povoleno, je navrácen kód 340, který indikuje, že zaslání článku do skupiny může být provedeno. Kód 440 říká, že zasílání do skupiny je z nějakých důvodů zakázáno.
Pokud je zasíláni dovoleno, článek by měl být prezentován ve formě specifikované v RFC 3977 [1] a měl by obsahovat všechny povinné řádky hlavičky. Potom co jsou hlavička i tělo zaslány serveru, server odpovídá kódem, kterým signalizuje úspěch nebo neúspěch datového přenosu.

Ihave

Syntaxe:

```
IHAVE message-id
```

Odpovědi:

Úvodní odpovědi:

```
335    Send article to be transferred
435    Article not wanted
436    Transfer not possible; try again later
```

Následující odpovědi:

```
235    Article transferred OK
436    Transfer failed; try again later
437    Transfer rejected; do not retry
```

Přítomnost tohoto příkazu je indikována schopností READER. Tento příkaz informuje server o skutečnosti, že klient má článek, jehož identifikátor je Message-ID. Pokud server požaduje kopii

tohoto článku, vrací odpověď, kterou klientovi sděluje, že má zaslat celý článek. Pokud server článek nechce, bude navrácena odpověď, že článek není serverem vyžadován.
Pokud je požadováno zaslání článku, měl by klient zaslat celý článek (zahrnující jak tělo tak hlavičku) na server. Server potom vrací odpověď s kódem indikující úspěch nebo neúspěch přenosu. Tento příkaz je určen pro podporu distribuce článků mezi servery.

Date

Syntaxe:

```
DATE
```

Odpovědi:

```
111 yyyymmddhhmmss    Server date and time
```

Přítomnost tohoto příkazu je indikována schopností READER. Příkaz slouží ke zjištění aktuálního UTC času na serveru.

Help

Syntaxe:

```
HELP
```

Odpovědi:

```
100    Help text follows (multi-line)
```

Tento příkaz je povinný. Poskytuje krátký soupis příkazů, které jsou daným serverem implementovány. Text nápovědy je presentován jako víceřádková textová odpověď.

Newgroups

Syntaxe:

```
NEWGROUPS date time [GMT]
```

Odpovědi:

```
231    List of new newsgroups follows (multi-line)
```

Přítomnost tohoto příkazu je indikována schopností READER. Po tomto příkazu bude zobrazen seznam skupin, vytvořený od data daného parametry <`date and time`>. Způsob zobrazení je stejný jako u příkazu LIST. Datum je zasláno jako 6 číslic ve formátu YYMMDD nebo YYYYMMDD. YY reps. YYYY jsou poslední dvě číslice roku, MM číslice udávající měsíc a DD je den měsíce. Pokud je potřeba, je číslo dne a měsíce uvedeno nulou. Při použití pouze dvou číslic pro určení roku se nejbližší století bere jako první část roku (např. 86 znamená 1986, 30 znamená 2030, 99 je 1999 a 00 je 2000).

Čas musí být také zadán. Musí být tvořen 6 číslicemi ve formátu HHMMSS (HH hodiny 0-23, MM minuty 00-59 a SS sekundy 00-59). Předpokládá se, že čas je uveden v časové zóně serveru. Pokud je uveden parametr *GMT*, jsou čas a datum vyhodnoceny vzhledem k greenwichskému času.

Newnews

Syntaxe:

```
NEWNEWS wildmat date time [GMT]
```

Odpovědi:

```
230    List of new articles follows (multi-line)
```

Přítomnost tohoto příkazu je indikována schopností NEWNEWS. Seznam identifikátorů zpráv zaslaných nebo doručených do diskusních skupin, jejichž název vyhovuje použitému *wildmat* řetězci, od doby dané parametrem *date* bude vypsán tak, že na každém řádku bude uveden právě jeden identifikátor zprávy. Poslední řádek bude obsahovat samostatnou tečku. Datum a čas jsou ve stejném formátu jako u příkazu NEWGROUPS.

List

Syntaxe:

```
LIST [keyword [wildmat|argument]]
```

Odpovědi:

```
215    Information follows (multi-line)
```

Přítomnost příkazu je indikována schopností LIST. Obecně tento příkaz slouží ke zjištění informací o diskusních skupinách. To o jaké informace se jedná je specifikováno parametrem *keyword* a to, kterých skupin se má tento příkaz týkat je omezeno parametrem *wildmat*.

Klíčová slova:

ACTIVE (může být následováno parametrem *wildmat)*

Navrací seznam obsažených diskusních skupin a k nim asociovaných informací. Každá diskusní skupina je zaslána jako řádek textu v následujícím formátu:

```
Group last first p
```

Kde `group` je jméno diskusní skupiny, `last` je číslo posledního aktuálně známého článku v diskusní skupině. `First` je číslo prvního článku, který je aktuálně ve skupině. A `p` je buď znak `y` nebo `n` a indikuje jestli je zasílání do dané diskusní skupiny povoleno (y) nebo zakázáno (n).

Hodnoty `first` a `last` jsou vždycky číselné, mohou být uvedeny nulami. Pokud je hodnota `last` menší než hodnota `first`, nejsou v dané diskusní skupině aktuálně žádné příspěvky.

Prázdný seznam zakončený tečkou na samostatném řádku je také možná odpověď, která signalizuje, že na serveru nejsou žádné diskusní skupiny.

ACTIVE.TIMES (může být následováno parametrem *wildmat*)
Navrací řádky skládající se z názvu diskusní skupiny, času vytvoření skupiny (udaném jako počet sekund od 1. ledna 1970), a emailové adresy uživatele zodpovědného za skupinu. Jednotlivá políčka jsou oddělena mezerou

DISTRIB.PATS (bez *wildmat* parametru)
Tento příkaz využívají klienti, aby zjistili jakou hodnotu vložit do hlavičkového řádku Distribution: u nově vytvářených článků. Odpověď na tento příkaz se skládá ze tří políček oddělených dvojtečkou. První políčko obsahuje váhu daného řádku, druhé políčko wildmat (pro rozlišení zda se má zkoumané skupiny týkat) a třetí políčko hodnotu, která se má použít pro pole DISTRIBUTION.

NEWSGROUPS (může být následováno parametrem *wildmat*)
Tento příkaz zobrazí pro zvolené skupiny řádky, na kterých je jméno diskusní skupiny a mezerou nebo tabelátorem oddělený krátký textový popis této skupiny.

OVERVIEW.FMT (bez *wildmat* parametru)
Vypíše pořadí a názvy jednotlivých položek v *overview* databázi (tj. některé hlavičkové hodnoty a metadata). Prvních pět řádků je povinných:

```
Subject:
From:
Date:
Message-ID:
References:
:bytes
:lines
```

Over

Syntaxe:

```
OVER message-id
OVER range
OVER
```

Odpovědi:

První tvar (podle Message-ID)

```
224    Overview information follows (multi-line)
430    No article with that message-id
```

Druhý tvar (podle rozsahu)

```
224    Overview information follows (multi-line)
412    No newsgroup selected
```

```
423    No articles in that range
```

Třetí tvar (podle aktuálního článku)

```
224    Overview information follows (multi-line)
412    No newsgroup selected
420    Current article number is invalid
```

Přítomnost tohoto příkazu je indikována schopností OVER. Příkaz vypíše hodnoty z overview databáze pro zadaný článek. Pro každý požadovaný článek vypíše řádek, kde je číslo článku , následované jednotlivými hodnotami, vše je oddělené tabelátory. Pořadí a typ hodnot, které jsou vypisované je možné zjistit příkazem LIST OVERVIEW.FMT. Příkaz over se buď může týkat právě jednoho článku specifikovaného zadáním Message-ID, nebo více článků určených parametrem *range*, který má stejný význam a syntaxi jako u příkazu LISTGROUP.

HDR

Syntaxe:

```
HDR field message-id
HDR field range
HDR field
```

Odpovědi:

První tvar (podle message-id)

```
225    Headers follow (multi-line)
430    No article with that message-id
```

Druhý tvar (podle rozsahu)

```
225    Headers follow (multi-line)
412    No newsgroup selected
423    No articles in that range
```

Třetí tvar (podle aktuálního článku)

```
225    Headers follow (multi-line)
412    No newsgroup selected
420    Current article number is invalid
```

Přítomnost tohoto příkazu je indikována schopností HDR. Tento příkaz umožňuje vypsat hodnotu pro požadovaný hlavičkový řádek z článku, který se buď určí pomocí Message-ID, nebo pomocí parametru *range* (stejný význam jako u OVER nebo LISTGROUP), nebo se bude jednat o aktuální článek ve vybrané skupině v případě, že článek není nijak jinak určen. Název hlavičkového řádku, jehož hodnotu chceme získat, se předá pomocí parametru *field*. Schopnost HDR udává též přítomnost příkazu LIST HEADER.

LIST HEADER

Syntaxe:

```
LIST HEADERS [MSGID|RANGE]
```

Odpovědi:

```
215    Field list follows (multi-line)
```

Tento příkaz vypíše názvy hlavičkových řádků získatelných prostřednictvím příkazu HDR. Použitelné názvy mohou být jiné pří selekci článku pomocí Message-ID nebo pomocí parametru *range*. Pro jakou variantu příkazu HDR chceme seznam použitelných názvů hlavičkových článků získat specifikujeme nepovinným parametrem.

Příkazy pro autentizaci

Autentizace vůči NNTP serveru je popsána v RFC 4643 [10]. V případě, že je použito zabezpečené spojení mezi NNTP klientem a serverem, může být autentizace provedena následujícím způsobem.

V případě, že je možno se k danému serveru přihlásit, musí server disponovat schopností AUTHINFO USER. Po úspěšném přihlášení uživatele, již tato schopnost nesmí být v seznamu schopností obsažena. Pro samotné přihlášení jsou k dispozici následující příkazy:

Syntaxe:

```
AUTHINFO USER username
AUTHINFO PASS password
```

Odpovědi:

```
281 Authentication accepted
381 Password required
481 Authentication failed/rejected
482 Authentication commands issued out of sequence
502 Command unavailable
```

Uživatel nejprve specifikuje prostřednictvím příkazu AUTHINFO USER své uživatelské jméno. Server vyzve uživatel k zadání hesla odpovědí se stavovým kódem 381. Uživatel zadá heslo ke svému účtu prostřednictvím příkazu AUTHINFO PASS. V případě, že je autentizace úspěšná, je navrácena odpověď s kódem 281, v opačném případě server navrací chybovou odpověď s kódem 481. Pokusí-li se uživatel zadat příkaz AUTHINFO USER, aniž by předchozím příkazem zadal své uživatelské jméno, je serverem navrácena odpověď s kódem 482. V případě, že je již uživatel úspěšně přihlášen, odpovídá server na další pokusy o přihlášení odpovědí se stavovým kódem 502.

Odpovědi NNTP Serveru

Existují dva druhy odpovědí, odpovědi vracející status a odpovědi textové.

Textová odpověď je zaslána pouze po numerické hodnotě udávající status a signalizující, že bude následovat text. Text je zaslán jako série řádků, ukončených dvojicí CRLF. Řádek obsahující

pouze tečku je použit k signalizaci konce textové odpovědi (jedná se o posloupnost znaků CRLF.CRLF). Z tohoto důvodu musí být tečky na samostatné řádku v těle zpráv zdvojeny a pak při interpretaci klientem opět uvedeny do původního stavu. Toto předzpracování a zapracování je v režii NNTP klientů.

Status obsažený v informacích udává odpověď na naposledy přijatý příkaz od klienta. Odpověď s informací o statusu se skládá z číselného kódu (3 číslice) a některé tyto odpovědi mohou obsahovat ještě textové upřesnění.

První číslice výstupu signalizuje úspěch, chybu, nebo proces zpracování předchozího příkazu:
1xx – Informační zpráva
2xx – Příkaz OK
3xx – Zatím je příkaz OK, zašlete jeho zbytek
4xx – Příkaz byl správně, ale z nějakého důvodu nemůže být proveden
5xx – Příkaz neimplementován nebo je nesprávný, nebo nastala nějaká vážná chyba v programu.

Další číslice v kódu indikuje kategorii odpovědi na funkci:
x0x – Spojení, nastavení
x1x – Výběr diskusní skupiny
x2x – Výběr článku
x3x – Distribuční funkce
x4x – Zasílání
x8x – Nestandardní (vyhrazené pro soukromou implementaci) rozšíření
x9x – Ladící výstup

Některé odpovědi obsahující status mohou obsahovat též parametry jako číslice a jména. Číslo a pozice takovýchto parametrů jsou stálé pro každý kód odpovědi. Parametry v odpovědích jsou od sebe a od kódu odpovědi odděleny jednou mezerou. Jedná-li se o číselné parametry jsou uvedeny v desítkové soustavě a mohou být uvedeny nulou.

Příklad dvou obecných odpovědí serveru:

```
200 server ready - posting allowed
502 access restriction or permission denied
```

Kdykoliv během sezení může server klientovi zaslat jednu z těchto obecných odpovědí:

- 500 – pokud příkaz není rozeznán, nebo není serverem implementován.

- 501 – pokud je chyba v syntaxi argumentů rozeznaného příkazu, nebo pokud je příkaz delší než je povoleno. Stejně tak musí být tento kód navrácen v případě, že základní příkaz je implementován, ale není implementována nějaká jeho nepovinná varianta.
- 403 – v případě, že došlo k interní chybě serveru.
- 480 – klient se musí autentizovat.
- 483 – klient musí založit bezpečnou komunikaci na již navázaném spojení (např. prostřednictvím SSL, nebo TLS).
- 401 – klient musí změnit stav spojení nějakým jiným způsobem.
- 400 – server ukončil z nějakého důvodu spojení s klientem.

Formát diskusních příspěvků

Formát diskusních příspěvků byl původně definován standardem RFC 850 [5], nyní je definován přímo v RFC 3977. Článek se skládá ze dvou částí – hlavičky a těla. Tyto části jsou odděleny pomocí jednoho prázdného řádku, jinými slovy pomocí dvou následujících párů CRLF. Aby článek splňoval obecné požadavky NNTP, článek nesmí obsahovat bajt s hodnotou NUL, nesmí obsahovat bajty s hodnotami LF a CR, pokud tyto netvoří dvojici CRLF. A musí končit dvojicí CRLF. Tato specifikace neklade žádné další omezení na tělo článku, může být i prázdné.

Hlavička článku se skládá z jednoho a více hlavičkových řádků. Každý takový řádek se skládá z názvu, dvojtečky, mezery, obsahu a dvojice CRLF v uvedeném pořadí. Jméno se skládá z jednoho nebo více tisknutelných ASCII znaků jiných než dvojtečka a pro účely RFC 3977 není citlivé na velikost písmen. Může existovat více hlavičkových řádků se stejným jménem. Obsah nesmí zahrnovat dvojici CRLF, ale může být prázdný. Hlavičkový řádek může být rozložen do více řádků, pokud se CRLF dvojce umístí před znak mezery nebo tabelátoru na řádku. Znaky CRLF určené pro rozřádkování nejsou považovány za obsah hlavičkového řádku. Obsah hlavičky by měl být v kódové sadě UTF-8.

Každý článek musí mít unikátní Message-ID, dva různé články na serveru nesmějí mít stejné Message-ID. Message-ID musí splňovat následující požadavky. Message-ID musí začínat znakem „<“ a končit znakem „>“. A tyto znaky nesmí obsahovat jinde než na konci. Message-ID. Musí mít délku 3 až 250 znaků. Message-ID se nesmí skládat z jiných znaků, než z tisknutelných US-ASCII znaků. Dvě Message-ID jsou stejná, pokud a jenom pokud se skládají ze stejné sekvence bajtů.

Popis HTTP protokolu

Úvod

Hypertext Transfer Protocol – HTTP/1.1 je definován v RFC 2616 vydaném v červenci roku 1999. V dnešní době je tato verze protokolu HTTP implementovaná ve všech dostupných prohlížečích. A obecně je protokol HTTP jedním z nejpoužívanějších protokolů v Internetu.

Jedná se o poměrně jednoduchý protokol. Komunikace je textová a probíhá metodou dotaz odpověď. Tento protokol pracuje s adresami URL a zpracovává informace různých typů tj. hypertextové informace (text, obraz, video, …). Pomocí odkazů vytváří orientovaný graf po jehož struktuře se uživatel pohybuje. Jedná se o bezstavový protokol, který má jen málo příkazů, které se zde nazývají metody. Standard MIME je využit pro zakódování dat přenášených http protokolem, což umožňuje přenášet téměř jakýkoliv typ dat. Jedná se o protokol typu klient/server . Komunikace mezi klientem a serverem probíhá pomocí protokolu TCP/IP a výchozím portem je port 80.

Jak už bylo řečeno HTTP je textový protokol, kde jednotlivé pole hlavičky mají obecně tvar `jméno pole : hodnota pole` a jsou ukončeny dvojicí znaků CR LF. Řádků hlavičky může být u různých zpráv různý počet. Za hlavičkou následuje prázdný řádek a pak vlastní tělo zprávy zakódované podle standartu MIME (informace o MIME kódování a typu aplikace jsou uvedeny v hlavičce). Rozlišujeme dva typy zpráv - požadavky a odpovědi

Požadavky

Zpráva s požadavkem z klienta na server obsahuje metodu, která bude aplikována na zdroj, identifikaci zdroje a verzi protokolu.

```
Request       = Request-Line
                *(( general-header
                 | request-header
                 | entity-header ) CRLF)
                CRLF
                [ message-body ]
```

Požadavek začíná řádkou *Request-line*, která začíná specifikací metody a za mezerou pokračuje URI zdroje a za další mezerou je verze protokolu HTTP.

```
Request-Line   = Method SP Request-URI SP HTTP-Version CRLF
```

Jednotlivé metody říkají co se má s danými daty na daný požadavek provést. Zde je seznam metod protokolu HTTP 1.1:

```
Method         = "OPTIONS"
                 | "GET"
                 | "HEAD"
                 | "POST"
                 | "PUT"
                 | "DELETE"
                 | "TRACE"
                 | "CONNECT"
```

```
                | extension-method
extension-method = token
```

Pro účely tohoto projektu stačí popsat pouze nejvýznamnější metody.

Nejprve se jedná o metodu GET. Příkaz GET zasílá klient a jeho provedení má za následek načtení dat specifikovaných adresou URL v požadavku ze serveru a jejich zaslání klientovi.

Další metodou je metoda HEAD, která se do značné míry podobá příkazu GET. Návratovou hodnotou po provedení tohoto příkazu jsou ale pouze meta informace o zdroji na dané URL.

Další metodou je metoda POST, která slouží pro zasílání údajů z klienta na server. Může se například jednat o data odeslaná pomocí formuláře na webové stránce. Data lze na server zaslat i pomocí metody GET a jejich začlenění do URL, ale v tomto případě je délka zasílaných dat omezena maximální délkou URL. U metody POST toto omezení není.

Metoda PUT již nepatří mezi povinně podporované příkazy, sloužila k odeslání dat z klienta na server. Metoda DELETE není také běžně podporovaná. Jejím účelem je vznesení požadavku o odstranění zadané URL.

Odpovědi

Po obdržení zprávy s požadavkem vrací server odpověď.

```
Response      = Status-Line *(( general-header
|response-header
                        | entity-header ) CRLF)
                       CRLF
                       [ message-body ]
```

Odpověď začíná řádkem Status-Line, dále následují řádky hlavičky. Každý řádek je ukončen dvojicí CR LF za prázdným řádkem pak může následovat tělo zprávy.

`Status-Line = HTTP-Version SP Status-Code SP Reason-Phrase CRLF`

Status-Line nejprve obsahuje verzi protokolu, mezeru, číselný kód odpovědi, další mezeru a doplňující textovou informaci. Protokol HTTP definuje množinu stavových kódů, jimiž dává najevo v jakém stavu se požadavek nachází. Stavový kód se skládá ze tří číslic, která stav identifikují a následného textového řetězce, který stav vysvětluje. Zde je seznam nejčastěji se vyskytujících návratových kódů:

Stavový kód	Textové vysvětlení
200 OK	Požadavek úspěšně splněn.
201 Created	Požadavek typu POST úspěšně vyřízen.
202 Accepted	Odpověď na asynchronní požadavek, potvrzení jeho správného přijetí.
204 No Content	Požadavek byl úspěšný, ale není co zobrazit.

300 Multiple Choices	Požadovaný dokument je dostupný z více míst. V odpovědi se vrací seznam možností. Pole Location odpovědi obsahuje možnost doporučenou serverem.
301 Moved Permanently	Požadovaná adresa URL byla přesunuta na jinou URL, která je obsažena v poli Locations odpovědi.
302 Moved Temporarily	Požadovaná adresa byla přesunuta dočasně na jinou adresu, ta je opět uvedena v poli Locations.
304 Not Modified	Dokument nebyl od doby definované polem If-Modified-Sience modifikován.
400 Bad Request	Server nerozumí požadavku.
401 Unauthorized	Požadavek musí být autentizován.
403 Forbiden	Požadavku nelze vyhovět.
404 Not Found	Zadané URL nebylo nalezena.
500 Internal Server Error	Vnitřní chyba serveru .
501 Not Implemented	Požadavek není implementován.
502 Bad Gateway	Brána obdržela od serveru neplatnou odpověď.
503 Service Unavailable	Server dočasně nemůže zpracovat požadavek (například z důvodu přetížení).

URL

URL (Uniform Ressource Locator) představuje konec šipky orientovaného grafu hypertextového prostředí. Obsahuje určení protokolu, IP adresy zdroje (často prostřednictvím plně kvalifikovaného doménového jména), číslo portu a přesnou lokaci požadovaného prostředku. Obsaženy mohou být další dodatečné informace, nejčastěji ve formě parametru pro interpret zpracovávající data. Příklad URL:
http://login.seznam.cz:80/?language=cz&remember=0&set_disableSSL=0&set_forceSSL=0&loggedURL=http%3A%2F%2Femail.seznam.cz%2Fticket&serviceId=email

Pole v hlavičkách

Obecná pole:

Tato pole se vyskytují nezávisle na tom jestli se jedná o požadavek nebo odpověď.

Connection	Určení voleb týkajících se spojení, token *close* signalizuje, že se jedná o poslední požadavek během persistentního spojení.
Date	Definice data a času vzniku zprávy

Pragma	Pole obsahující direktivy závislé na konkrétní implementaci
Transfer-Encoding	Určuje metodu kódování aplikovanou na přenášená data

Pole v požadavcích

Accept	Typy standartu MIME, které budou akceptované v odpovědích. Je možné použít zástupný znak hvězdičky
Accept-Charset	Akceptovatelné znakové sady. Kromě implicitních.
Accept-Encoding	Obsahuje množinu přijatelných hodnot pole Content-Encoding v odpovědi. Hodnota je definovaná standardem MIME
Accept-Language	Omezuje množinu akceptovaných jazyků v odpovědi
Authorization	Pro použití serveru, které nedovolují anonymní přístup ke zdrojům. Zasílají se též informace o oprávnění uživatele.
From	E-mailová adresa uživatele, který odeslal požadavek
Host	Specifikuje adresu a port serveru na kterém klient žádá o data
If-Modified-Since	Modifikátor příkazů GET. GET načte pouze data v případě, že byla od zadaného časového údaje modifikována
Referer	Určuje URL z něhož získal klient požadované URL
User-Agent	Jméno a verze klienta HTTP

Pole v odpovědích

Age	Sděluje zasilateli očekávané množství času od doby, kdy byla odpověď vygenerována na původním serveru.
Location	Využití tohoto pole je popsáno výše.
Proxy-Authenticate	Toto pole musí být součástí odpovědi s kódem 407 a hodnota obsahuje autentikační schéma a parametry aplikovatelné na Proxy-Server.
Etag	Poskytuje konkrétní hodnotu entity pro požadovanou variantu.
Retry-After	Pole definuje časový interval, po který nebudou požadované služby k dispozici.
Server	Jméno a verze serveru HTTP.
Accept-Ranges	Umožňuje serveru zjistit jeho akceptování určitého rozsahu požadavku na daný zdroj
WWW-Authenticate	Pole se používá pro přístup s jednoduchou autentizací typu výzva/odpověď. Informace o uživatelském oprávnění se nešifrují

Spojení:

HTTP/1.1 server by měl předpokládat, že HTTP/1.1 klient bude chtít udržovat persistentní spojení, dokud nezašle řádek hlavičky *Connection* s hodnotou *close*. Pokud server uzavře spojení ihned po obsloužení požadavku, měl by vybavit řádek hlavičky *Connection* hodnotou *close.*

Pokud klient nebo server zašlou hodnotu *close* uvnitř řádku hlavičky *Connection*, požadavek se stává posledním požadavkem pro dané spojení.

Služby systému Windows NT

Služba NT je proces běžící na pozadí, který je spuštěn Service Control Managerem jádra systému NT. Obvykle je služba spuštěna po startu systému, ještě před tím, než se nějaký uživatel přihlásí k systému a běží nezávisle na přihlášení různých uživatelů k systému. Není-li služba spuštěna automaticky při startu systému, může být spuštěna uživatelem buďto přes ovládací panel Služby, nebo prostřednictvím jiného programu, který obsahuje rozhraní pro ovládání Service Control Manageru.
Jedním z mnoha typů programů, který přímo vybízí k implementaci v podobě služby systému NT jsou síťové servery, u nichž často vyžadujeme nepřetržitý běh, který není nijak ovlivněn přihlašováním a odhlašováním uživatelů k systému.

Služby jsou normální 32 bitové aplikace pro systém Windows a mohou být buď vybaveny grafickým uživatelským rozhraním (s *WinMain()* funkcí) nebo mohou být pouze konzolové (s *main()* funkcí). Hlavní funkce služby by neměla dělat nic, pouze zaregistrovat servisní *ServiceMain()* (tento název je dán konvencí, ale může být i jiný) funkci, zavoláním *StartServiceCtrlDispatcher()* a skončit. Systém Windows NT bude potom volat zaregistrovanou funkci *ServiceMain()* při požadavku na spuštění služby.

ServiceMain() by potom měla spustit hlavní vlákno, které aktuálně provádí kód vlastní služby. Po tomto se však funkce *ServiceMain()* nesmí ukončit do té doby, než je služba zastavena. V případě, že je potřeba službu restartovat, Service Control Manager jednoduše znovu zavolá funkci *ServiceMain().*

Funkce *Handler()* přejímá řídící požadavky od Service Control Manageru. Zde je uvedena standardní množina požadavků, kterou používá SCM k řízení služby:

- Stop (SERVICE_CONTROL_STOP): Přikáže službě, aby se zastavila.
- Pause (SERVICE_CONTROL_PAUSE): Přikáže službě, aby provedla operaci pauza.
- Continue (SERVICE_CONTROL_CONTINUE): Přikáže službě, aby obnovila svůj běh po pauze.
- Interrogate (SERVICE_CONTROL_INTERROGATE): Požaduje zjištění aktuálního stavu služby
- ShutDown (SERVICE_CONTROL_SHUTDOWN): Signalizuje službě, že nastává vypnutí systému

Funkce *Handler()* pak příchozí požadavky obsluhuje různými způsoby. Požadavky Pause a Continue jsou obslouženy jednoduchým pozastavením a opětovným spuštěním hlavního vlákna. Požadavky ShutDown a Stop většinou vedou k nastavení sdílené proměnné, která se testuje v hlavním vláknu a v případě, že její stav signalizuje konec, je hlavní jádro ukončeno a taktéž je následně ukončeno vlákno *ServiceMain()*. Požadavek *Interrogate* je obsloužen aktualizací informace o stavu služby. Tato aktualizace je provedena i při jakékoliv jiné změně stavu. Stavy, ve kterých se služba může nacházet, jsou následující:

- SERVICE_START_PENDING
- SERVICE_RUNNING
- SERVICE_PAUSE_PENDING
- SERVICE_PAUSED
- SERVICE_CONTINUE_PENDING
- SERVICE_STOP_PENDING

Službě mohou být též předány uživatelsky definované řídící požadavky a jakékoliv potřebné parametry mohou být umístěny v systémovém registru a to v oblasti vyhrazené pro danou službu. Každá služba může vytvářet podklíče a hodnoty ve svém klíči:

HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Services\<ServiceName>

Kde Servicename je jméno služby registrované v ControlManageru.

Relační databáze

Úvod

Relační databáze (popř. relační databáze s podporou objektů) jsou v dnešní době nejpoužívanější databázovými systémy. V roce 1969 přišel doktor E. F. Codd se svou představou o databázi založené na matematickém aparátu relačních množin a predikátové logiky. Databázová relace se od matematické poněkud liší. Má zavedený pomocný aparát nazvaný schéma relace. Schéma relace říká, jaký je název relace, kolik má atributů a jaké jsou jejich názvy a domény (doména určuje přípustné hodnoty pro daný atribut). V databázích je schématem relace definice struktury tabulky. Jednotlivé atributy tvoří sloupce tabulky. Každý sloupec má definován jednoznačný název, typ a rozsah (doménu). Záznam se stává n-ticí (řádkem) tabulky. Pokud jsou v různých tabulkách sloupce stejného typu, pak tyto sloupce mohou vytvářet vazby mezi jednotlivými tabulkami. Tabulky se poté naplňují vlastním obsahem - konkrétními daty. Ovšem relací není jenom tabulka, ale cokoliv strukturovaného do řádků a sloupců. Což znamená, že relací je i výsledek jakéhokoliv dotazu, a tak s ním můžeme pracovat. Kolekce více tabulek, jejich funkčních vztahů, indexů a dalších součástí tvoří relační databázi.

Cizí klíč je sloupec databázové tabulky, který odkazuje na jiný sloupec (jiné tabulky nebo i stejné tabulky). Hodnoty takového sloupce musí být shodné s některou z hodnot ve sloupci, ke kterému je klíčem. Vytváří se tak reference – odkaz. Podmínka shody se kontroluje při všech operacích nad databází, což se označuje jako referenční integrita.

Primární klíč je pole nebo kombinace atributů, jednoznačně identifikující každý řádek v databázové tabulce. Žádný atribut, který je součástí primárního klíče, nesmí obsahovat nedefinovanou hodnotu. Každá tabulka může mít definovaný pouze jeden primární klíč. Primární klíč musí mít následující vlastnosti: jedinečnost, neměnnost, nenulovou hodnotu. Typickým příkladem primárního klíče je rodné číslo v seznamu osob, identifikační číslo v seznamu podniků apod.

SQL

SQL je standardizovaný dotazovací jazyk používaný pro práci s daty v relačních databázích. SQL je zkratka anglických slov Structured Query Language (strukturovaný dotazovací jazyk). SQL příkazy se typicky dělí do čtyř základních skupin.

1. Příkazy pro manipulaci s daty – jedná se o příkazy určené pro získávání a modifikaci dat. Označují se jako DML – Data Manipulation Language (jazyk pro manipulaci s daty).
2. Příkazy pro definici dat – jedná se o příkazy určené k vytváření jednotlivých struktur databáze (tabulek, indexů, pohledů, …). Některé struktury lze také upravovat. Tato skupina příkazů se nazývá DDL – Data Definition Language (jazyk pro definici dat).
3. Příkazy pro řízení databáze – jedná se o příkazy určené k nastavování přístupových práv a řízení transakcí. Označují se jako DCL – Data Control Language, někdy také TCC – Transmission Control Commands.
4. Ostatní příkazy jsou určeny pro správu databáze. Pomocí nich lze přidávat uživatele a nastavovat systémové parametry. Tato skupina není standardizována a konkrétní syntaxe příkazů závisí na použitém databázovém systému.

Pro modelování struktury relační databáze se používají ER diagramy.

Regulární výrazy

Regulární výrazy budou v tomto projektu využity především pro lexikální analýzu, tj. pro zjišťování, zda příkazy zaslané na server splňují formu předepsanou standardem. A dále budou použity pro získání podřetězců z řetězců s příkazem, např. pro získání konkrétních parametrů zaslaného příkazu.
Formální definice regulárních výrazů je následující:

Regulární výrazy nad konečnou abecedou Σ a regulární množiny, které označují, jsou rekurzívně definovány takto:

I. ε je regulární výraz označující regulární množinu { ε }

II. Ø je regulární výraz označující regulární množinu Ø
III. *a* je regulární výraz označující regulární množinu {*a*} pro všechny *a* z Σ,
IV. jsou-li *p*, *q* regulární výrazy označující regulární množiny *P* a *Q*, pak
    a. (*p* + *q*) je regulární výraz označující regulární množinu *P* ∪ *Q*,
    b. (*pq*) je regulární výraz označující regulární množinu *P.Q*,
    c. (*p**) je regulární výraz označující regulární množinu *P**.
V. Žádné jiné regulární výrazy nad ∑ neexistují.

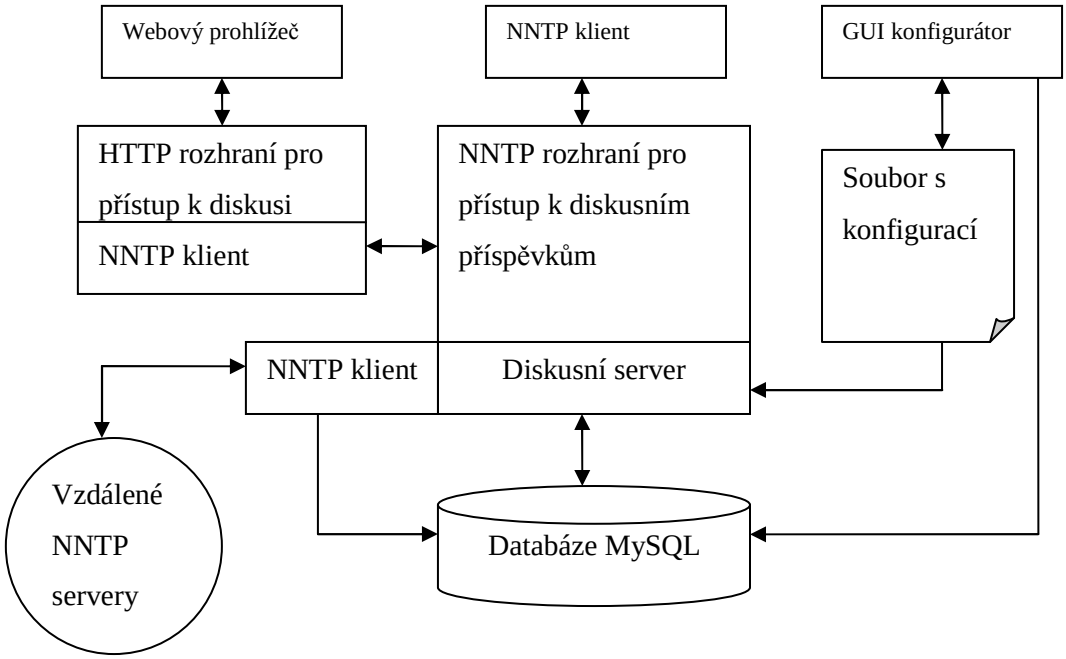
Při praktickém použití se obvykle využívají rozšířené definice regulárních výrazů, které umožňují běžné konstrukce zapsat jednodušším způsobem (ale schopnosti takto rozšířených výrazů se od základní definice neliší). Nejčastější jsou následující konstrukce:

- Namísto znaku + (plus) se pro alternativy používá znak |
- *Výběr z uvedených znaků*: zápis *[a-z0-9A-Z]* znamená libovolný znak v rozsahu *a–z, 0–9,A-Z* (tedy alfanumerický znak).
- *Jeden z neuvedených znaků*: zápis *[^a-z0-9A-Z]* znamená libovolný znak mimo znaků v rozsahu *a–z, 0–9,A-Z.* V některých implementacích se místo stříšky používá znak ! (vykřičník).
- *Libovolný znak*: symbol . (tečka) znamená jakýkoliv libovolný znak.
- *Vymezení začátku řetězce*: zápis *^nějakývýraz* popisuje pouze takové řetězce *nějakývýraz*, které se nachází na začátku řádku.
- *Vymezení konce řetězce:* zápis *nějakývýraz$* popisuje pouze takové řetězce *nějakývýraz*, které se nachází na konci před koncem řádku.
- *Volitelná část:* zápis *xy?* znamená, že znak *x* může, ale nemusí, být následován znakem *y*.
- *Libovolný počet opakování, ale nejméně jednou:* zápis *x*+ vyžaduje (narozdíl od *x**) alespoň jeden výskyt znaku *x* (a je ekvivalentní k zápisu *xx**).
- *Určený počet opakování:* zápis *x{3,6}* popisuje, že znak *x* se musí opakovat minimálně třikrát a maximálně šestkrát.
- *Definované množiny znaků:* některé speciální skupiny znaků mají zkratkový zápis, např. *d* pro libovolnou desítkovou číslici, \s pro bílý znak apod.

Návrh systému

Návrh struktury aplikace

Struktura aplikace je znázorněna na obr. 1.



Obr. 1: Struktura aplikace

Základem je databázový systém, pro ukládání všech potřebných persistentních informací, kterými jsou diskusní skupiny, diskusní články a s nimi asociované údaje. Vytvářený NNTP server je ve skutečnosti program poskytující síťové rozhraní s příkazy NNTP protokolu pro přístup k databázi. Druhým programem je HTTP Server, který má v sobě integrovaného NNTP klienta. Tento server převádí HTTP požadavky na NNTP příkazy a zasílá je NNTP serveru. Příchozí odpovědi navrací jako HTML stránky. Takto zvolená struktura aplikace navíc umožňuje mít server umístěný na jiném počítači, než je umístěn databázový systém a stejně tak je umožněno, aby nad danou databází operovalo více serverů. Uživatel tak může využít pro přístup k diskusním příspěvkům specializovaného NNTP klienta (Mozilla Thunderbird, Outlook Express) nebo webový prohlížeč. Dále umožňuje, aby HTTP rozhraní zobrazovalo články z nějakého cizího NNTP serveru, pokud je lokální NNTP server v proxy módu. Protože bývá problém s grafickým rozhranním programu, který běží jako služba NT, je konfigurátor oddělená aplikace s GUI, která pouze vygeneruje konfigurační soubor. Tento konfigurační soubor si načte služba při startu, resp. při restartu. Konfigurační program obsahuje též grafické rozhranní k databázi, pomocí kterého se dá provádět administrace (přidávání/ odebírání skupin, článků, uživatelů, ...)

NNTP server navíc obsahuje NNTP klienta, který se spouští v určitých časových okamžicích a který stahuje nové články a diskusní skupiny ze vzdálených diskusních serverů a ukládá je do lokální databáze.

Návrh NNTP rozhraní

NNTP rozhraní bude reprezentované vlastním programem, ve kterém se nejprve načtou konfigurační informace. Ty obsahují informace o tom, na kterém portu má NNTP server naslouchat a zda bude pracovat s lokální databází a nebo bude přeposílat požadavky na vzdálený NNTP server tj. bude v módu proxy. V případě proxy módu jsou IP adresa a port vzdáleného NNTP serveru obsaženy také v množině konfiguračních informací.

Nejprve dojde k vytvoření serverového socketu pro protokol TCP/IP, který může být volitelně zabezpečený. Následně se zahájí naslouchání na daném socketu. V cyklu se bude testovat, zda se nevyskytlo nějaké příchozí spojení. Pokud tomu tak je, tak se toto spojení akceptuje. Dále se vytvoří a spustí obsluhující vláknu a tomuto vláknu se jako parametr předá socket s příchozím spojením a jeho úkolem je příchozí spojení obsloužit. Funkce reprezentující obslužné vlákno budou dvě: jedna se bude používat pro obsluhu proxy módu a druhá pro obsluhu práce s lokální databází.

Při práci se socketem je také možnost použití časovače, který vyprší v případě, že po nastavenou dobu nedojde k žádné vstupně výstupní operaci na daném socketu. V takovém případě bude klientovi zaslána zpráva `Connection Timeout` a spojení bude uzavřeno.

Funkce reprezentující vlákno použité k zajištění komunikace v proxy módu bude vypadat následovně. Po akceptování příchozího spojení server vytvoří klientský socket, pomocí něhož se připojí na vzdálený NNTP diskusní server. IP adresa tohoto serveru a port služby budou obsaženy v množině konfiguračních informací. Po úspěšném připojení na vzdálený server budou data mezi klientským a serverovým socketem přenášena následovně. Data načtená z klientského socketu se zapíší do serverového socketu a data načtená z klientského socketu budou zapsána do serverového socketu. Do dat nebude programem nijak zasahováno, tj. nebudou nijak upravována ani filtrována. Stejně tak jsou mezi oběma sockety propagována uzavření socketů. Tj. v případě že je uzavřen jeden ze socketů, je následně uzavřen i ten druhý.

Funkce reprezentující vlákno obsluhující požadavky na lokální databázi s diskusními příspěvky bude vypadat asi následovně. V případě, že funkce očekává na vstupu příkaz NNTP protokolu, bude maximální délka přijímaných dat omezena na 512 znaků. Při překročení této velikosti bude zobrazena chybová zpráva. Pro každý příchozí příkaz bude zjištěno, zda vyhovuje některému z regulárních výrazů, které budou nastaveny tak, aby pokrývaly syntaxi příkazů NNTP standardu. Pokud příkaz nevyhovuje žádnému z regulárních výrazů, jedná se o nepodporovaný nebo chybně zapsaný příkaz a o této skutečnosti je opět uživateli podáno chybové hlášení. Pomocí regulárních výrazů se také ze syntakticky dobře zapsaných příkazů extrahují parametry. U některých složitých příkazů jako např. příkaz NEWNEWS bude potřeba použít i víceúrovňovou extrakci parametrů prostřednictví regulárních výrazů. Na základě toho o jaký dotaz se jednalo a jaké parametry byly zadány, se vytvoří příkaz SQL. V některých případech je potřeba transformovat NNTP příkaz na sekvenci SQL dotazů. Takto formulované dotazy jsou zaslány modulu, který zprostředkovává komunikaci mezi serverovým

rozhraním a databázovým systémem. Tento modul provede databázové dotazy a navrátí data získaná jako výsledek těchto dotazů. Ta potom budou serverovým rozhraním transformována na odpovědi vyhovující NNTP standadtu. Tyto odpovědi budou zaslány pomoci socketu klientovi. Mírně odlišné je zpracováni příkazu POST, kdy se nejprve načtou textová data, jejichž maximální délka je omezena konfiguračními nastaveními, a teprve po načtení těchto dat je zkonstruován SQL příkaz, kterým se data zapíší do databáze. V případě, že je databázový systém nedostupný je uživateli sděleno informační hlášení o vnitřní chybě programu.

Návrh HTTP rozhraní

Síťový podsystém HTTP rozhraní je stejný jako u rozhraní NNTP, tj. opět existuje hlavní serverové vlákno inicializované podle konfigurační nastavení, které spouští jednotlivá vlákna obsluhující příchozí požadavky. Funkce, kterou je reprezentováno vlákno obsluhující příchozí požadavky bude pracovat následujícím způsobem. Funkce bude očekávat požadavek založený na metodě GET nebo POST a na protokolu HTTP/1.1 nebo HTTP/1.0. Pokud budou tato dvě kritéria splněna, z URL požadavku se vypreparuje požadovaný cíl. Na základě toho, na jaký imaginární cíl požadavek odkazuje se zavolá příslušná funkce. Nejprve se očekává, že klient bude chtít zobrazit úvodní stránku tedy cílem požadavku bude „/“. Server na základě tohoto požadavku vygeneruje HTML stránku, která bude obsahovat seznam diskusních skupin na serveru. Seznam skupin a další informace se vždy získají prostřednictvím integrovaného NNTP klienta, který se připojí na lokální NNTP server (zabezpečeným nebo nezabezpečeným spojením), prostřednictvím příkazů NNTP protokolu získá požadované informace a následně je transformuje na HTML kód. Ve výše uvedeném seznamu diskusních skupin bude každá položka odkazem na seznam článků ve zvolené diskusní skupině, který bude organizován podobně jako seznam diskusních skupin, ale místo názvů skupin v něm budou uvedeny předměty, autoři a data jednotlivých článků. Každá položka tohoto seznamu bude odkazem na článek jejíž předmět ji tvoří. V seznamu bude implementováno řazení a odsazení jednotlivých řádků tak, aby odpovídalo hierarchii odpovědí mezi články. Součástí stránky bude též odkaz k zobrazení formuláře pro zaslání nového článku a odkaz pro návrat na předchozí stránku.

Pokud je požadováno zobrazení nějakého článku, je odkaz na něj specifikován dvěma parametry a to pomocí identifikátoru diskusní skupiny a číslem článku v diskusní skupině.. V případě, že je požadováno zobrazení článku, je vygenerována speciální stránka, na které bude zobrazen odesílatel (obsah hlavičky *From*), datum (obsah hlavičky *Date*) a předmět (obsah hlavičky *Subject*) článku. Dále bude následovat tělo článku. Součástí každé stránky se zobrazeným článkem budou odkazy na následující článek v dané diskusní skupině (pokud existuje), na předcházející článek v dané skupině (pokud existuje), odkaz pro odpověď na článek a odkaz pro návrat na předchozí stránku. V případě, že se nepodaří dané URL převést na nějaké volání interní funkce, nebo parametry zadané jako součást

URL nebudou správné, např. budou odkazovat na neexistující článek apod., bude vrácena odpověď `404 Not Found`.

V případě, že bude server NNTP vyžadovat autorizaci, bude zobrazeno webovým klientem okno pro zadání uživatelského jména a hesla a tyto autentizační údaje budou zaslány NNTP serveru.

Návrh konfiguračního rozhraní

Bude se jednat o samostatný program, který zobrazí grafické uživatelské rozhranní pro zadání potřebných konfiguračních informací. Po zadání těchto informací uživatelem program zkontroluje správnost zadaných údajů a v případě, že je vše v pořádku, uloží zadané informace do souboru. Pokud je v údajích detekována nějaká chyba, bude uživateli zobrazeno chybové hlášení.

Druhá část tohoto rozhranní bude zprostředkovávat grafické zobrazení informací uložených v databázi a jejich administraci, kterou se myslí administrace uživatelů, diskusních skupin, článků, distribučních vzorů a seznamu serverů pro vzdálené stahování diskusních příspěvků.

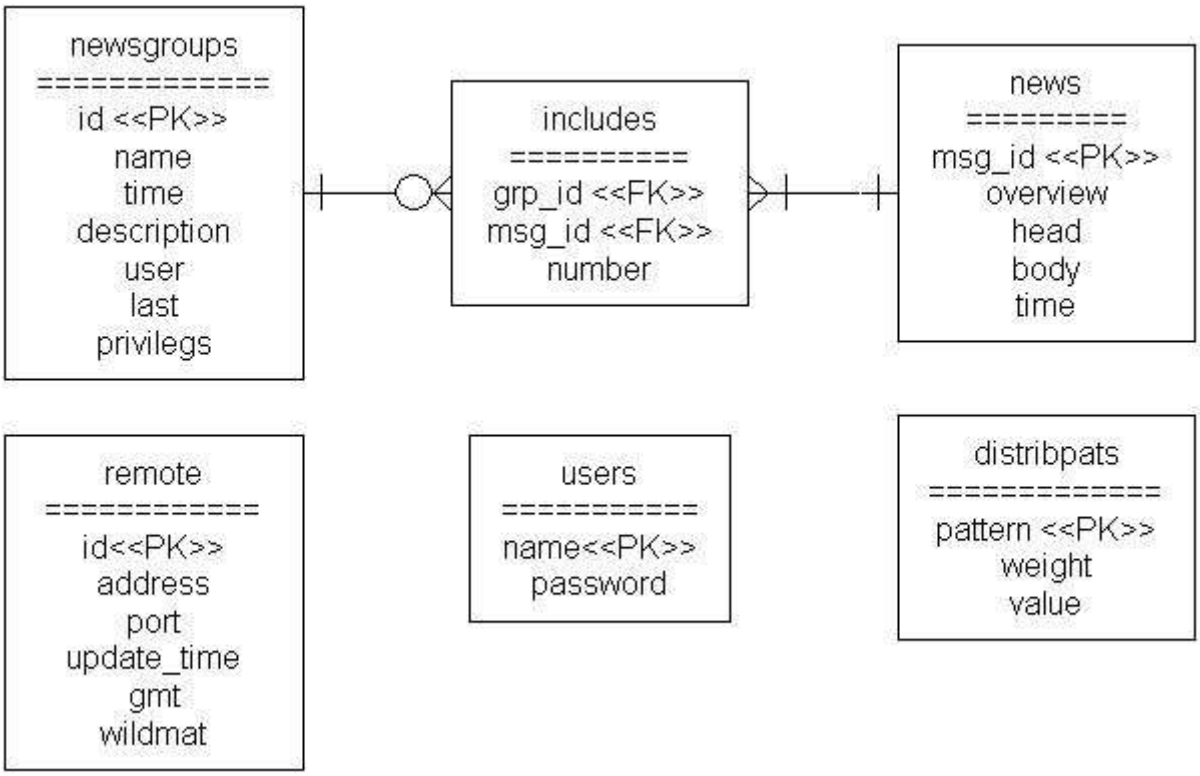
Návrh automatizovaného NNTP klienta

Základem vlákna zajišťujícího toto distribuční rozhraní bude časovač, který jednou za nastavený čas spustí NNTP klienta. Tento NNTP klient bude mít za úkol z databáze načíst IP adresy vzdálených NNTP serverů, na tyto adresy se připojit a postahovat nové články z diskusních skupin. *Wildmats* řetězce pro vymezení názvů požadovaných diskusních skupin budou taktéž v databázi. Po provedení těchto kroků se NNTP klient ukončí.

Přesněji bude běh NNTP klienta fungovat následovně. Nejprve se načte seznam IP adres vzdálených NNTP serverů a seznam diskusních skupin pro aktualizaci. Potom se NNTP klient začne v cyklu postupně připojovat prostřednictvím socketů k jednotlivým serverům. Z databáze si vyzvedne čas poslední aktualizace článků (tj. čas svého posledního spuštění) a následně serveru zašle příkaz NEWGROUPS pro zjištění, zda od té doby nebyly na serveru vytvořeny nějaké nové diskusní skupiny. Pokud tomu tak je, vytvoří ve své databázi nové diskusní skupiny. Následně zašle příkaz NEWNEWS s parametrem *wildmats* určujícím názvy požadovaný skupin, které hodlá aktualizovat, a datem poslední aktualizace. V případě, že v požadovaných diskusních skupinách jsou nějaké nové články, vrátí server v odpovědi seznam identifikátorů těchto článků. Pomocí příkazu ARTICLE <Message-ID> postahuje klient jednotlivé články z diskusního serveru a prostřednictvím SQL příkazů je vloží do databáze. Duplikátní články do databáze nebudou vloženy, neboť tomuto vložení zamezí integritní omezení jedinečnosti pro atribut identifikátor článku. Tento proces NNTP klient provede pro všechny diskusní servery, které má v seznamu. Následovat musí nové uložení času poslední aktualizace do souboru a následné ukončení vlákna s NNTP klientem.

Návrh databáze

Pro zvolenou aplikaci bude použit databázový systém MySQL, ve kterém bude vytvořeno šest tabulek pro uložení informací o diskusních skupinách, o jednotlivých článcích, o náležitostech článků ke skupinám, o uživatelských účtech, účtech vzdálených NNTP serverů a o distribučních vzorech. Pro



bližší popis je zde umístěn ER diagram (obr. 2).

Obr. 2: ER diagram

Implementace systému

Projekt je implementován v programovacím jazyce Java a ke komunikaci s databázovým prostředím MySQL využívá rozhraní JDBC (ovladač Connector/J 5.0.4). Jazyk Java byl vybrán, protože je volně dostupný a disponuje množstvím tříd (Socket, URL, URI, String, ...), které umožňují rychlou a pohodlnou implementaci síťových služeb. Dalším důvodem pro výběr tohoto jazyka je automatická správa paměti, přenositelnost a větší systémová bezpečnost. Tyto aspekty jsou dány tím, že programy v Javě jsou překládány do bajtkódu, který je při spuštění interpretován Java Virtual Machine (JVM). Tento mechanismus si však vybírá svou daň před v podobě vyšší paměťové a procesorové náročnosti aplikací.

V následujících kapitolách je popis implementace jednotlivých rozhraní aplikace, jejich dekompozice do tříd a popis funkčnosti jednotlivých tříd a jejich vzájemné komunikace. Pro snazší pochopení částí týkajících se databázových transakcí je uvedena kapitola s popisy použitých databázových tabulek systému MySQL. Obsažena je také kapitola věnující se problému užití Java aplikací jako služeb NT.

Implementace NNTP rozhraní

Server je implementovaný tak, aby zcela vyhovoval RFC3977. Server umožňuje zabezpečený a nezabezpečený přístup. Dále je možné přepínat mezi lokálním a *proxy* módem. Pokud NNTP server poskytuje zabezpečené připojení, je možná autentizace uživatele. Administrátor může také určit dobu vypršení spojení, maximální velikost zasílaných článků, název serveru, mód kompatibility se starým RFC a jeho rozšířeními, zakázání zasílání článků, zapnutí výpisů logů, zapnutí synchronizace s jinými NNTP servery (nevyžadujícími autentizaci) a nastavení intervalu této synchronizace. Vytvořený server disponuje příkazy spadajícími pod následující seznam schopností (*capabilities*): READER, POST, OVER, NEWNEWS, AUTHINFO USER, LIST ACTIVE, LIST ACTIVE.TIMES, LIST DISTRIB.PATS, LIST NEWSGROUPS, LIST OVERVIEW.FMT . Programový kód implementující NNTP rozhranní je rozdělen do tříd NNTPServer, SocketHandler, NNTPprocesor, NNTPProxy, Head, Update. Následovat bude popis jednotlivých tříd, jejich funkce a způsob jejich propojení a komunikace.

Popis třídy NNTPServer

Tato třída obsahuje metodu *public static void main(String args)*, kterou se spouští celý NNTP server. Metoda *main()* se pokusí načíst ze souboru objekt s konfigurací, tento objekt je instancí třídy *Configuration*. Pokud se tento objekt nepodaří načíst, vytvoří se instance nového objektu s konfigurací s výchozím nastavením. Následuje inicializace statických proměnných se vzory regulárních výrazů ve třídě *NNTPprocessor* voláním metody *NNTPprocesor.initializeRE()* a dále zavedení databázového ovladače voláním *NNTPprocesor.initializeJDBC()*. Pokud se některá z těchto inicializačních metod nepodaří, program se ukončí.

Pokud je to konfigurací vyžadováno, dojde k vytvoření zabezpečeného serverového socketu, k tomu je potřeba, aby v konfiguračním objektu byla obsažena cesta k souboru s klíčem a heslo k tomuto souboru. Pokud zabezpečený server není požadovaný, vytvoří se standardní serverový socket, reprezentovaný instancí třídy *Socket*. Tento socket se sváže s požadovaným číslem portu (získané z konfiguračního objektu) a zahájí se na něm naslouchání. Příchozí požadavky jsou akceptovány a pro jejich obsluhu se vytvoří vlákno, které je buď instancí třídy *SocketHandler* v případě, že je požadována práce s lokálními složkami, nebo je instancí třídy *NNTPProxy*, pokud je

požadováno přeposílání požadavků na vzdálené diskusní servery. Konstruktoru zvoleného vlákna se předá odkaz na objekt s konfigurací a odkaz na objekt třídy *Socket* s navázaným klientským spojením. V této úvodní části se též spouští vlákno, které je instancí třídy *Update* a které zajišťuje stahování příspěvků ze vzdálených NNTP serverů v určitých časových intervalech.

Popis třídy SocketHandler

Konstruktor této třídy si uloží do lokálních proměnných konfigurační informace, které vezme z konfiguračního objektu předaného jako parametr konstruktoru. V metodě *run()* této třídy se nejprve vytvoří instance třídy NNTPProcesor(), jejíž metody slouží ke zpracování příchozích NNTP příkazů. Dále se z klientského socketu vytvoří vstupní a výstupní proudy. Vstupní i výstupní proudy jsou dva. První dvojice čte data ve znakové sadě UTF-8 a slouží pro načítání příkazů a výpisů těch odpovědí serveru, které neobsahují data z článků a jejich součástí.. Druhá dvojice čte data ve formátu ISO-8859-1 a slouží pro načítání a výpis těl a hlaviček článků. Kódová sada UTF-8 může reprezentovat jeden znak určitou sekvencí několika bajtů, kdežto u kódování ISO-8859-1 každý bajt odpovídá jednomu znaku.

Na základě toho, jestli je v konfiguraci povolené zasílání článků na diskusní server, zašle server klientovi příslušnou uvítací zprávu. Následuje cyklus, ve kterém se nejprve testuje, jestli objekt třídy *NNTPProcesor* nepožaduje ukončení cyklu čtení a zpracování příkazů z důvodu zaslání příkaz *QUIT*. To se zjistí voláním metody *int getStatus()* výše uvedeného objektu. Pokud tomu tak není, zjistí se jestli navrácená hodnota neurčuje, že se místo příkazu mají zaslat data článku, tj. že předchozím příkazem byl příkaz *POST*. Pokud tomu tak bylo, čte se ze socketu článek až po koncovou sekvenci *CRLF.CRLF*. A načtená data se předají instanci třídy *NNTPProcesor* pomocí metody *postArticle(dataBuffer)* ke zpracování. V případě, že není požadováno načítání článku, čte se příkaz voláním lokální metody *readCommand(commandReader)*. Návratová hodnota této metody signalizuje, zda byl načten příkaz správně, nebo zda při čtení došlo k chybě, nebo jestli nebyl zadán příliš dlouhý příkaz. V takovém případě, je klientovi zaslána zpráva *501 Command too long*. V dalším kroku se klientskému socketu nastaví časovač, jehož hodnota je získána také z konfigurace. Pokud během nastaveného časového intervalu nedojde k žádné operaci se socketem, generuje se výjimka *SocketTimeoutException*, která vede k uzavření socketu z důvodu vypršení času relace. Načtený příkaz se předá objektu třídy *NNTPProcesor* ke zpracování prostřednictvím metody *processCommand*(*String command*). Pomocí metody *getOutput()* stejného objektu se načte odpověď na zadaný příkaz (popř. na zaslaná data článku) a dále se pomocí metody *getSetOutputType()* zjistí, jaký výstupní proud má být použit (UTF-8 nebo ISO-8859-1) a vynuluje se interní označení tohoto proudu pro příští příkaz. Do vybraného výstupního proudu se zašle odpověď.

Popis třídy NNTPProxy

Tato třída implementuje vlákno, které obsluhuje klientský socket v případě, že je požadován *proxy* mód. Konstruktorem se předají všechna potřebná konfigurační nastavení, kterými jsou port vzdáleného NNTP serveru, adresa tohoto serveru, cesta k souboru s klíčem a heslo k tomuto souboru v případě, že je požadováno zabezpečené připojení na vzdálený NNTP server. Samozřejmostí je také předání ukazatele na Socket s příchozím spojením.

Problém s proxy módem v Javě je ten, že neexistuje způsob, jak zjistit, jestli je dané spojení opravdu aktivní. Jinými slovy, nedají se detekovat situace, když druhá strana uzavře spojení, nebo když dojde např. k fyzickému přerušení spojení). Jediným způsobem jak se dá takováto situace detekovat, je zápis nebo čtení ze socketu, které v takovém případě skončí výjimkou. Tato technika je z hlediska proxy serveru nepoužitelná, protože nechceme, aby do probíhající komunikace bylo nějakým způsobem zasahováno. Ale nemůžeme situaci takto nechat, aby zbytečně nezůstávaly otevřené sockety a aktivní vlákna, která obsluhují porušené spojení. Je totiž potřeba detekovat uzavření jednoho ze socketů a na základě této události uzavřít i socket druhý.

Řešení proxy módu ve zde popisovaném systému je následující. Nejprve se vytvoří objekty reprezentující vstupní a výstupní proudy pro oba sockety. Následuje „nekonečný“ cyklus, ve kterém se postupně otestuje jeden a druhý socket, jestli některý z nich neobsahuje data, a pokud ano, tak se data načtou ze vstupního proudu jednoho socketu a zapíší do výstupního proudu druhého socketu. Pokud k takovéto výměně dojde, vynuluje se interní čítač. V každé iteraci cyklu se vlákno na určitý čas uspí voláním metody *sleep()* a po obnovení běhu vlákna se zvýší hodnota interního čítače. Pokud hodnota interního čítače dosáhne předem definované hodnoty, cyklus se ukončí a oba dva sockety se zavřou. Tj. k uzavření spojení dojde po určité době nečinnosti. Druhá možnost, kdy se cyklus ukončí a dojde k uzavření obou socketů je, že během některé operace se socketem dojde k výjimce. Jinými slovy, vlákno vytváří určité sezení v proxy módu, které může vypršet v případě delší nečinnosti, i když druhá strana tj. vzdálený NNTP server si ještě připojení ukončit nepřeje (např. doba sezení zde ještě nevypršela).

Popis třídy Head

Objekt této třídy poskytuje funkce pro zpracování hlavičky diskusního článku a získání informací v ní obsažených. Konstruktor této třídy očekává jediný parametr, kterým je hlavička diskusního článku v podobě textového řetězce. Metoda *boolean check()* slouží k otestování, zda hlavička splňuje všechny požadavky na její formát, dané v RFC3977. Metoda *check* využívá regulárních výrazů sestavených přesně podle syntaktických pravidel uvedených v onom RFC. Hlavní metodou, která zpracování hlavičky provede, je metoda *buildOverview(int bytes, int lines)*, která vytvoří na základě hodnot v hlavičce řetězec, který je později vrácen serverem jako odpověď na příkaz *OVER*. Tj. vypreparuje

z hlavičky hodnoty pro hlavičkové řádky *Subject*, *From*, *Date*, *Message-ID*, *References* a doplní počet bajtů těla článku a počet řádků těla článku. Tyto údaje jsou předány parametry při volání metody *buildOverview*. Mezi jednotlivé položky umístí znak tabelátor a výsledný řetězec uloží do proměnné *output*. Obsah této proměnné je možné získat voláním *getOutput()*. Před tímto voláním je však nutné zavolat metodu *getErr()*, kterou se zjistí, zda při zpracování nedošlo k nějakým chybám. Pokud v hlavičce není obsažen hlavičkový řádek Message-ID, je nové Message-ID vygenerováno na základě jména NNTP serveru (předáno konfiguračním objektem) a aktuálního času. Vygenerované Message-ID pro daný objekt třídy *Head* lze získat voláním *getMessageID()*.

Druhá důležitá metoda této třídy je metoda *LinkedList getNewsGroups(Connection conn)*. Úkolem této metody je zjistit obsah hlavičkového řádku *Newsgroups*, vybrat z něj názvy jednotlivých diskusních skupin a s využitím databázového připojení (*Connection*) zjistit pomocí SQL dotazu, zda jsou uvedené skupiny obsažené v lokální databázi. Názvy skupin, které jsou obsažené v lokální databázi jsou vloženy do seznamu (*LinkedList*), který tato metoda navrací. Než jsou však názvy skupin použity v SQL dotazu, musejí být z kódování ISO-8859-1, ve kterém jsou jako součásti hlavičky načteny, převedeny do kódování UTF-8, ve kterém jsou uloženy v databázi. K tomuto překódování slouží metoda *private String encodeString(String str)*.

Popis třídy Update

Tato třída implementuje vlákno zajišťují stahování diskusních příspěvků ze vzdálených NNTP serverů. V databázi je uložen seznam serverů, tedy přesněji pro každý vzdálený NNTP server je tam uložena jeho adresa, port, *wildmat* řetězec (vymezuje názvy skupin, jichž se bude synchronizace týkat) a časové razítko (udává čas poslední aktualizace) a informace o tom zda byl použit lokální nebo GMT čas.

Konstruktorem se této třídě předají konfigurační informace, kterými jsou řetězec pro připojení k NNTP databázi (př.: *jdbc:mysql://localhost/nntp?user=user_name&password=password& useUnicode=true &characterEncoding=utf8*) a dále časový interval po kterém se má aktualizace spouštět. Dále metoda provádí v cyklu následující operace. Nejprve načte z databáze adresu a port vzdáleného NNTP serveru a pomocí objektu *Socket* se k němu připojí. Následně vzdálenému NNTP serveru zašle příkaz *CAPABILITIES*. Pokud server odpoví na tento příkaz kódem 500, tj. že zadaný příkaz nezná, jedná se o server implementovaný podle starší verze NNTP protokolu. Pokud server na příkaz *CAPABILITIES* odpoví kódem 101 a v seznamu schopností serveru je schopnost *READER* a *NEWNEWS*, může začít stahování nových příspěvků. V případě, že v seznamu schopností není schopnost *READER*, ale *MODE-READER*, jedná se o *mode-switching* server a je potřeba serveru zaslat příkaz *MODE READER* pro přepnutí módu a otestovat dostupnost potřebných schopností opětovným zasláním příkazu *CAPABILITIES*. Pokud nejsou schopnosti *READER* a *NEWNEWS* dostupné, spojení s tímto serverem se ukončí. V případě, že se jedná o server implementovaný podle

RFC977, zašle se pro jistotu příkaz *MODE READER* a to, že požadované příkazy nejsou dostupné se zjistí až když budou použity a server navrátí příslušnou chybovou odpověď.

Následuje zaslání příkazu *NEWGROUPS* následovaného časovým razítkem převedeným ve tvaru *YYYYMMDD HHmmss* a tokenem *GMT* v případě, že se nejedná o lokální, ale GMT čas. Výsledkem tohoto dotazu je víceřádková odpověď, která obsahuje seznam nově vytvořených skupin od zadaného data. Z každého řádku se vyjme jen první parametr, kterým je název diskusní skupiny a ten se uloží do dynamického pole reprezentovaného objektem třídy *Vector*. Následně se pomocí iterátoru objekt třídy *Vector* prochází a pomocí příkazu *LIST NEWSGROUP název_skupiny* se program pokouší pro každou skupinu získat její popis, který je třetím parametrem v odpovědi na výše uvedený příkaz. Pokud se serveru podaří popis skupiny získat, vytvoří v lokální databázi pomocí příkazu jazyka SQL novou prázdnou skupinu s oním popisem.

Dále klient zašle serveru příkaz *NEWNEWS wildmat YYYYMMSS HHmmss* [*GMT*], tento příkaz by měl vypsat Message-ID všech nových zpráv vytvořený od zadaného data a jejichž skupina vyhovuje uvedenému *wildmat* řetězci. Každý řádek této odpovědi (tj. Message-ID) se uloží do dynamického pole reprezentovaného objektem třídy *Vector*. Toto pole se iterátorem prochází a pro každé Message-ID se zašle příkaz *ARTICLE <Message-ID>* pro doručení požadovaného článku. Tento článek se předá metodě *postArticle(String Article)*, která je naprosto stejná jako u třídy *NNTPProcesor* a která zajistí kontrolu všech náležitostí článku, vygenerování hodnot pro *overview* databázi a uložení článku se všemi náležitostmi do databáze.

Po stažení všech článků se spojení se serverem ukončí a aktualizuje se časové razítko v databázovém záznamu týkající se daného vzdáleného NNTP serveru. Po stažení příspěvků ze všech v databázi obsažených vzdálených serverů se vlákno uspí na dobu získanou z konfiguračního objektu. Po probuzení vlákna se znovu provedou všechny výše uvedené kroky.

Popis třídy NNTPProcesor.java

Třída NNTPProcesor zajišťuje vlastní zpracování NNTP příkazů. Kromě proměnných sloužících k uložení konfigurace z konfiguračního objektu, který je předán jako parametr konstruktoru, obsahuje tato třída důležité proměnné activeNewsgroup a activeArticle, pomocí kterých jsou implementovány vnitřní ukazatele na aktuálně nastavenou skupinu a článek.

Třída obsahuje dvě statické metody, první *initializeJDBC*() zajišťuje nahrání databázového ovladače, druhá *initializeRE*() zajišťuje inicializaci vzorů (*Pattern*) pro vytváření regulárních výrazů. Tyto vzory jsou bezpečné z hlediska vláken a jejich kompilaci tímto způsobem stačí provést jen jednou před tím, než se z dané třídy začnou vytvářet její instance.

Funkce *processCommand(String command)* zajišťuje vlastní zpracování NNTP příkazu předaného jako textový řetězec v parametru *command*. Funkce nejprve provede převod došlého příkazu na velká písmena, protože všechny regulární výrazy jsou vystavěny nad velkými písmeny.

Jako první se provede porovnání došlého příkazu s regulárním výrazem, který zkontroluje, zda příkaz obsahuje na začátku řetězec některého z implementovaných výrazů. Pokud tomu tak není, do proměnné *output*, jejíž obsah je posléze zasílán klientovi se zapíše chybové hlášení *500 Unknown command*. Pokud je základní příkaz znám, postupně se testuje na shodu s regulárními výrazy, které již kontrolují, jestli jsou i jeho parametry ve správném tvaru. V případě, že základ příkazu byl správně, ale nepodaří se nalézt shodu s podrobným regulárním výrazem pro daný příkaz, je zapsáno hlášení *501 Syntax error*.

Zpracování příkazu AUTHINFO

Nejprve se porovná došlý příkaz na shodu s regulárním výrazem pro příkaz *AUTHINFO*. Pokud dojde ke shodě, musí se nejprve zjistit, jestli je server v zabezpečeném stavu tj. jestli je použito šifrování na transportní vrstvě. Není-li, je uživateli zaslána zpráva *483 Datastream is insufficiently secure* a autentizace není povolena. Je-li bezpečné spojení navázáno, nejprve se otestuje jestli už byl zaslán příkaz *AUTHINFO USER* s uživatelským jménem, což je označeno pravdivostní proměnnou. V případě, že tomu tak není a uživatel nyní použil příkaz AUTHINFO s parametrem PASS, je navrácena zpráva *482 Authentication commands issued out of sequence*. Je-li nyní specifikováno uživatelské jméno, uloží se a poznačí se do pravdivostní proměnné tato skutečnost, stejně tak je zaslána uživateli zpráva *381 Password required*. Po zaslání obou příkazů tj. po předání uživatelského jména i hesla, se na řetězec s heslem zavolá metoda *hashPassword(String password)*, kterou se vytvoří MD5 hash hesla. V databázi se totiž neukládá přímo heslo, ale jeho MD5 hash. Řetězec s zahashovaným heslem a uživatelským jménem se předá metodě *makeAuth(String user, String hashedPassword)*. Tato metoda se připojí k databázovému serveru a SQL dotazem `SELECT name FROM users WHERE name=? AND pass=?` zjistí, zda je daný uživatel v databázi s daným heslem asociován. Ke všem dotazům na databázi v tomto programu se používá objekt třídy *PreparedStatement*, pomocí kterého je bezpečné sestavovat SQL dotazy, protože sám zajistí ohlídání taktiky SQL injection, kdy se uživatel zadáním neregulárního vstupu snaží získat z databáze citlivé a nedovolené údaje. Je-li záznam v databázi nalezen, proběhla autentizace úspěšně a metoda *makeAuth()* vrací pravdivostní hodnotu *true*. Na základě té je vygenerována NNTP odpověď *281 Authentication accepted*, vrátí-li metoda *makeAuth() false*, návratová zpráva bude *481 Authentication failed/rejected*. Dojde-li během zpracování např. během operace s databází k nějakému problému, je v tomto případě ostatně jako i v celém dalším kódu uživateli zobrazena chybová zpráva *403 internal program error*. Po úspěšné autentizaci se nastaví pravdivostní proměnná *userAuthenticated*, což zamezí při výpisu schopností (příkaz *CAPABILITIES*) vypisovat schopnost *AUTHINFO USER* a také při dalším pokusu o přihlášení zobrazí zprávu *502 User already authenticated*.

Zpracování příkazů ARTICLE, STAT, BODY, HEAD

Další testovaným příkazem je příkaz *ARTICLE*, *STAT*, *BODY*, *HEAD*, který se testuje reg. výrazem, pomocí kterého se z příkazu získají dva řetězce. Prvním je příkazové slovo a druhým parametr, který nemusí být obsažen, nebo je jím číslo udávající článek, nebo Message-ID článku. Tyto dva řetězce se předají funkci *article()*, která provede zpracování. Předtím, než se však parametry příkazu předají funkci pro zpracování (a to platí pro všechny dále zpracovávané příkazy) zavolá se funkce *authRequired()*, která zjistí zda je požadována autentizace a v případě, že ano, vrací zprávu *480 Authentication required* a zastaví zpracování aktuálního příkazu.

Metoda *article()* tedy nejprve zjistí, zda je požadována specifikace článku pomocí číselného parametru nebo Message-ID. V případě, že je založena na číselném parametru, musí být již vybrána nějaká aktivní skupina, jejíž *id* je v proměnné *activeNewsgroup*. V případě, že vybrána není, je v *activeNewsgroup* hodnota -1 a uživateli je zobrazeno chybové hlášení *412 no newsgroup has been selected*. Není-li zadán parametr žádný, předpokládá se použití ukazatele na aktuální článek reprezentovaného proměnnou *activeArticle*. Pokud je tento nastaven na hodnotu -1, není zvolen žádný aktuální článek a je zasláno chybové hlášení *420 no current article has been selected*. Podle základního příkazu a metody specifikace článku se vytvoří dotaz nad databází. Příklad tohoto dotazu pro příkaz *ARTICLE* a specifikaci pomocí čísla článku je `SELECT number, news.msg_id, head, body FROM news, includes WHERE news.msg_id = includes.msg_id AND includes.number=? AND includes.grp_id=?`. Kombinace příkazů a metod specifikace článku vede na osm různých dotazů. V případě, že proběhlá transakce vrátí výsledek, je tento předán do proměnné *output* jako odpověď protokolu NNTP na daný příkaz při zvolené metodě určení článku.

Pokud po provedení dotazu, kdy určení článku záviselo na aktuálně nastavené skupině popř. aktuálně nastaveném článku, nejsou navrácena databází očekávaná data, je proveden test, ve kterém se zjistí, zda-li v databázi opravdu existuje aktuálně nastavená skupina popř. aktuální článek. Tyto údaje mohli být během NNTP relace smazány administrátorem, nebo nějakým jiným způsobem znepřístupněny a klient je o této skutečnosti informován chybových hlášení s kódem 412,420 nebo *423 no such article number in this group* v případě, že aktuální skupina existuje, ale uživatelem zadané číslo článku je chybné.

Zpracování příkazu LIST

Dalším příkazem, který se zpracovává, je příkaz *LIST*. Pro tento příkaz existují dvě verze regulárních výrazů. První testuje varianty příkazu *LIST*, u kterých není povolen parametr *wildmat*. Jedná se tedy o příkaz *LIST* bez parametrů a příkaz *LIST* s parametrem *DISTRIB.PATS* nebo *OVERVIEW.FMT*. Pokud je zjištěna verze příkazu *LIST* bez parametru, zavolá se metoda *list(String wildmat)* s parametrem *null*. Popis této metody bude uveden později.

Je-li zjištěna varianta *OVERVIEW.FMT*, vypíše se jako odpověď obsah řetězce, který obsahuje uspořádání položek v *OVERVIEW* databázi.

U varianty *DISTRIB.PATS* dojde k zavolání funkce *listDistribPats()*. Tato metoda pouze získá z databáze celý obsah tabulky *distripats*, mezi jednotlivé položky každého řádku získaného z databáze vloží dvojtečky a tyto řádky vrací jako víceřádkovou odpověď NNTP serveru na výše uvedený příkaz.

U variant příkazu *LIST*, které mohou být následovány nepovinným parametrem *wildmat* se postupuje takto. Zjistí-li se varianta příkazu *ACTIVE.TIMES* volá se metoda *listActiveTimes()*, jejímž parametrem je zadaný *wildmat* řetězec. Uvnitř této metody se nejprve zavolá metoda *wildmatProcessing(wildmat)*, která na základě zadaného *wildmatu* vrátí část databázového dotazu, která se přidá za klauzuli *WHERE* SQL příkazu pro zjištění požadovaných informací a ovlivní tím rozsah příkazu pouze na skupiny, které vytyčuje *wildmat* řetězec. Metoda *activeTimes()* pomocí takto připraveného SQL příkazu zjistí název, čas a adresu uživatele zodpovědného za vytvoření dané skupiny. Časové razítko převede na sekundy a tyto hodnoty naformátuje tak, aby odpovídali standardu NNTP.

Použití klíčového slova *ACTIVE* má za následek vyvolání metody *list(wildmat)*. Tato metoda provede stejné zpracování parametru jak je uvedeno výše, vygenerovanou část dotazu přidá za klauzuli *WHERE* příkazu pro zjištění jména skupiny, maximálního a minimálního čísla článku v dané skupině a zjištění, zda je do skupiny umožněno zasílání. Tento příkaz je celkem komplikovaný, skládá se ze dvou poddotazů a musí být ošetřeno, že minimální a maximální číslo článku u prázdné skupiny nesmí být null, ale číslice nula. Zjištěné řádky s odpověďmi jsou naformátovány tak, aby konformovali s NNTP protokolem.

U varianty příkazu *LIST* s klíčovým slovem *NEWSGROUPS* probíhá zpracování podobně jako u předchozích variant, je však prováděno metodou *listNewsgroups()* a informace, která se zjišťuje, je název a popis diskusní skupiny. Základem všech tří výše uvedených metod je, jak již bylo napsáno, metoda *wildmatProcessing*(*String wildmat*). Tato metoda v cyklu zpracovává každý *wildmat_pattern*, ze kterého se skládá předaný *wildmat* řetězec. Využije se funkce *LIKE* jazyka MySQL, která má svůj formát parametru *wildmat*, kdy místo znaku * slouží k označení sekvence znaků znak *%* (procento) a k označení právě jednoho znaku, znak _ (podtržítko). Proto je potřeba v zadaném *wildmat_pattern* tyto znaky neutralizovat tím, že se před ně vloží znak \ (zpětné lomítko) a následně znak * nahradit znakem % a znak ? znakem _. V případě, že *wildmat_pattern* začíná znakem ! (vykřičník) tj. je požadována negace vyhledávání, použije se MySQL funkce *UNLIKE*. Při zpracování se musí postupovat odzadu a musí se zajistit spojení negace porovnání s předchozími porovnáními logickou spojkou AND, kladné porovnání s předchozími porovnáními spojkou OR a pomocí závorek zajistit zpracování tak, aby vyhovovalo RFC 3977. Příklad převodu dvou *wildmat* řetězců na MySQL podmínku jsou:

```
a*,!*c,d* -> (E like d%) OR ( E NOT LIKE %c AND ( E LIKE a%))
a*,c?d,d* -> (E like d%) OR ( E LIKE c_d OR (E LIKE a%))
```

Zpracování příkazu GROUP

Dalším zpracovávaným příkazem je příkaz *GROUP*. Tento příkaz má jediný parametr, kterým je jméno skupiny. To se předá metodě *group(String groupName)*. Tato metoda se podobným SQL příkazem, který byl použit v metodě *list()* pokusí zjistit maximální a minimální číslo článku ve specifikované skupině a *id* této skupiny. Pokud je provedení příkazu úspěšné, vloží se získané *id* do proměnné *activeNewsgroup* a do *activeArticle* se vloží číslo prvního článku (tj. minimální číslo článku) ve skupině. Uživateli je v tomto případě zobrazena zpráva s kódem 211. Při nenalezení zadané skupiny v databázi je navrácena zpráva *411 no such newsgroup*.

Zpracování příkazu LISTGROUP

Následuje test na příkaz *LISTGROUP*. Tento příkaz může být následován nepovinným názvem skupiny a nepovinným parametrem *range*. Pro zpracování parametru *range* je použit následující reg. výraz: `((\\d{1,16})(-|-(\\d{1,16}))?)?)?`, pomocí něj se zjistí o jakou variantu parametru range se jedná (existují 3 různé varianty). Metodě *listGroup()* se předá jméno diskusní skupiny a informace týkající se parametru *range*. V případě, že této metodě nebyl předán název skupiny, zjistí zda je nastavena aktuální skupina a pokud tomu tak není, nebo pokud nastavená aktivní skupina v databázi reálně neexistuje, je navrácena zpráva *411 no such newsgroup*. V opačném případě, nebo při specifikaci skupiny parametrem, se složí MySQL dotaz, jehož úkolem je získat nejprve max. a min. číslo článku. A další SQL dotaz, jehož úkolem je získat čísla článků, ležící mezi hodnotami specifikovanými parametrem *range*. V jedné transakci jsou oba tyto příkazy provedeny a výsledek zobrazen v souladu s NNTP protokolem. V případě, že byla pomocí parametru specifikována diskusní skupina a databázová operace byla úspěšná, dojde k přenastavení ukazatelů na aktuální článek a aktuální skupinu.

Zpracování příkazu NEWGROUPS

Příkaz *NEWGROUPS* má dva povinné parametry, kterými jsou čas a datum a jeden nepovinný parametr, který udává, zda se jedná o GMT údaje. Všechny tyto údaje jsou zpracovány prostřednictvím metody *newgroups(Matcher mStat)*. Nejprve se zjistí, který ze dvou možných formátů data byl použit, všechny číselné údaje se složí do jednoho řetězce, který se doplní řetězcem udávajícím zónu, která může být buď *GMT-00:00* nebo výchozí časová zóna systému, na kterém je server spuštěn. Výsledný řetězec je tedy ve tvaru *yyyyMMddHHmmssz* nebo *yyyyMMddHHmmss*. Tento řetězec se převede na časové razítko, které je následně využito v rámci SQL dotazu k nalezení těch záznamů v tabulce *newsgroups*, jejichž časové razítko je novějšího data, než to získané z příkazu. V rámci uvedeného příkazu se musejí též získat minimální a maximální číslo článků z nalezené skupiny. Získané údaje se použijí k sestavení NNTP odpovědi, která je svým formátem stejná jako u příkazu *LIST* nebo *LIST ACTIVE*.

Zpracování příkazu NEWNEWS

Následuje zpracování příkazu *NEWNEWS*. Tento příkaz má navíc oproti příkazu *NEWGROUPS* parametr *wildmat* udávající, které skupiny mají být prohledávány na přítomnost nového článku. Zpracování příkazu je v režii metody *newnews(Matcher mStat)*, která zpracuje zadané údaje naprosto stejným způsobem jako metoda NEWGROUPS, tj. převede je na časové razítko. Dále zavolá metodu *wildmatProcessing()* na zadaný *wildmat* řetězec, která je popsána u příkazu *LIST*. Následuje složení SQL příkazu, který získá Message-ID zpráv, jež byly vytvořeny dříve, než udává získané časové razítko. K tomuto příkazu se připojí omezující SQL kód, získaný prostřednictvím příkazu *wildmatProcessing()*. Po provedení takto zkomponovaného příkazu se vypíší získaná Message-ID v souladu s NNTP standardem.

Zpracování příkazu OVER

Příkaz *OVER* je zde uvedenou implementací podporován pouze v doporučené variantě s parametrem *range*. Při použití příkazu *OVER* s parametrem *Message-ID* je vypsáno hlášení *503 this over variant not implemented*. V konfiguraci je možné zvolit, aby místo klíčového slova *OVER* fungovalo slovo *XOVER*, které je využíváno nynějšími implementacemi NNTP klientů a patří mezi jedno z rozšíření původního protokolu RFC 977. Zpracování parametru *range* je provedeno stejným způsobem jako u příkazu *LISTGROUP*. Informace o parametru *range* jsou předány metodě *over(long param1, long param2)*, která nejprve zjistí, jestli existuje nějaká aktuální skupina podle obsahu proměnné *actualNewsgroup*, pokud tomu tak není, vypíše "*412 no newsgroup current selected*", pokud není zadán žádný parametr a je předpokládána volba článku daného ukazatelem aktuálního článku, je tento ukazatel otestován a v případě, že aktuální článek není nastaven je navrácena odpověď *420 No article selected*. Následuje složení SQL příkazu, jehož úkolem je získání čísla článku a řetězce s položkou *overview* z databáze. Rozsah čísel článků, o nichž budou tyto údaje z databáze získány je omezený hodnotami parametru *range*. Po úspěšném provedení databázové operace je pro každý získaný řádek spojeno číslo článku s řetězcem z položky *overview* pomocí znaku tabelátor a takto vzniklý řetězec je navrácen jako NNTP odpověď. Pokud nebyla SQL příkazem získána, žádná data, testuje se, jestli náhodou nebyla během NNTP relace smazána aktuální diskusní skupina, nebo aktuální článek. V takovém případě je zobrazeno příslušné chybové hlášení, jinak je vypsáno chybové hlášení *423 Empty range*, udávající, že byl zadán invalidní rozsah.

Zpracování příkazu POST

Při zjištění, že je zadán příkaz POST, se na základě toho, zda je na server povoleno zasílání nebo ne, vygeneruje buď zpráva *340 Input article - end with <CR-LF>.<CR-LF>* nebo zpráva *440 Posting not permitted*. V prvním případě, je do proměnné *status* nastavena hodnota, která nadřazenému vláknu (instanci *SocketHandler*) signalizuje, že nyní má načíst tělo článku a ne příkaz. Po načtení těla článku je proměnná udávající jaký druh informace se má číst nastavena opět tak, aby signalizovala čtení

příkazu. Tělo článku je nadřazeným objektem předáno metodě *postArticle(String article)*, kdy parametrem je textový řetězec s daty článku, jehož uložení do databáze je požadováno.

Metoda *postArticle()* nejprve zjistí, jestli tělo článku neobsahuje některé nepovolené znaky. Pokud tomu tak je, je navrácena chybová zpráva *441 Posting failed - invalid article format*. Dále se nalezne v textu článku první prázdný řádek, který odděluje hlavičku od těla, a článek se rozdělí právě na tyto dvě části. Následuje vytvoření instance objektu *Head* pro hlavičku právě zpracovávaného článku. V další části se spočítá počet řádků těla článku (počet výskytů párů *CRLF*) a počet bajtů článku (ten je rovný počtu znaků). Tyto údaje se předají metodě *buildOverview()* náležející k dříve vytvořené instanci třídy *Head*. Taktéž se zavolá metoda *getNewsGroups(conn)*, která pro danou hlavičku navrátí seznam s názvy skupin, které jsou spravovány lokální databází. Pokud některá z těchto metod skončí neúspěšně, je opět navrácen chybový kód 411. V opačném případě se příkazem *getOverview*() získá *overview* položka pro daný článek, příkazem *getMessageID()* Mesage-ID daného článku a pomocí *getHead()* zpracovaná popř. upravená hlavička článku. Tyto informace a řetězec s tělem článku se použijí pro zkomponování SQL příkazu, který tyto údaje vloží do tabulky *news*. V databázi se pomocí SQL příkazu *LOCK TABLE* uzamknou tabulky *newsgroups*, *news* a *includes* pro zápis. Vloží se článek se všemi údaji do tabulky *news*. Následně se pro každou skupinu ze seznamu získaném příkazem *getNewsGroups()* zvedne čítač maximálního čísla článku skupiny v tabulce *newsgroups*, a do tabulky *includes* se vloží Message-ID článku, id skupiny a číslo článku, které je v daný okamžik právě oním maximálním číslem článku pro danou skupinu.

Po zpracování všech skupin, do kterých měl být článek zaslán se vypíše *240 article posted ok*. V případě, že se do databáze nepodařilo článek vložit, je buď zobrazena zpráva *441 posting failed - posted Message-ID still exists*, pokud je to z důvodu toho, že článek s daným Message-ID již existuje, nebo zpráva *403 program error, function not performed*, pokud je to z důvodu jiného.

Zpracování ostatních příkazů

Příkaz *CAPABILITIES* pouze vypíše schopnosti serveru. V případě, že je server v bezpečném módu a je vyžadována autentizace je tento seznam schopností doplněn o položku *AUTHINFO USER*. Tato položka je ze seznamu schopností odebrána po úspěšném přihlášení.uživatele.

Příkaz *HELP* pouze zobrazí textový popis všech možných příkazů NNTP serveru.

Příkaz *DATE* zobrazí aktuální GMT čas ve formátu předepsaném NNTP standardem.

Příkaz *QUIT* vloží do proměnné *output* řetězec: *205 Connection closing* a nastaví pomocnou proměnnou status na -1, čímž zajistí ukončení hlavního cyklu zpracování příkazů v instanci třídy *SocketHandler* a následné ukončení spojení s klientem.

Při zpracování příkazu *LAST* a *NEXT* je volána metoda *lastNext()* s parametrem udávajícím, ze kterého příkazu je volána. Tato metoda nejprve ověří existenci aktuální skupiny a aktuálního článku, popř. vypíše chybová hlášení. Je-li vše v pořádku, pokračuje metoda aplikací SQL příkazu, kterým se u příkazu *NEXT* zjistí číslo následujícího článku a u příkazu *LAST* číslo předchozího článku. Pro

zjištěné číslo článku se dalším SQL dotazem pokusí z databáze získat Message-ID daného článku. Pokud se mu to povede, nastaví zjištěné číslo článku do proměnné *actualArticle* udávající aktuální článek a navrátí odpovídající NNTP odpověď s kódem 223.

Implementace HTTP rozhranní

Vytvořený webový server poskytuje zabezpečený (HTTPS) nebo nezabezpečený (HTTP) přístup. Dále je podporován výběr diskusních skupin k prohlížení, prohlížení seznamu článků v dané skupině a prohlížení jednotlivých článků. Do skupiny lze prostřednictvím tohoto rozhraní zaslat nový příspěvek, nebo reagovat na příspěvek existující. Spojení rozhraní s NNTP serverem může být taktéž zabezpečené nebo nezabezpečené. Přes webové rozhraní může být provedena i autentizace uživatele vůči NNTP serveru, pokud si autentizaci NNTP server vyžádá. Kód implementující HTTP rozhranní je rozdělen do tříd HTTPServer, NNTPClient, SocketHandler, Mime, Article, ArticleBuild, RequestProcessor, Overview. Následovat bude popis jednotlivých tříd, jejich funkce a způsob jejich propojení a komunikace.

Popis třídy HTTPServer

Tato třída obsahuje metodu *public static void main(String[] args)).* Tedy tato třída bude použita pro spuštění celého programu. Úkolem této třídy je nejprve ze souboru načíst objekt, který je instancí třídy *Config*. Některé informace z tohoto objektu budou následně využity i ostatními spolupracujícími objekty. Dále třída vytvoří serverový socket (reprezentovaný objektem *Socket*) naslouchající na portu, jehož číslo je získáno metodou *getHttpPort()* z konfiguračního objektu . V případě, že je vyžadován zabezpečený HTTP přístup tj. protokol HTTPS, vytvoří se zabezpečený serverový socket pomocí mechanismu SSLv3. Pro vytvoření takovéhoto zabezpečeného socketu je potřeba programu navíc předat název souboru s klíčem (*KeyStoreFile*) a heslo k tomuto souboru (*KeyStorePassword*). V případě, že je se serverovým socketem navázáno spojení, spustí se vlákno, které bude toto spojení obsluhovat. Vlákno je instancí třídy *SocketHandler*.

Popis třídy SocketHandler

Pomocí konstruktoru této třídy se nové instanci obslužného vlákna předá objekt reprezentující socket s klientským připojením a dále informace s konfigurací. Metoda *run()* této třídy má nejprve za úkol vytvořit instanci třídy *RequestProcessor*, která slouží k syntaktické a sémantické analýze HTTP požadavků, a následně čte ze socketového vstupu jednotlivé řádky HTTP požadavku. Konec vstupu (požadavku) je signalizován načtením prázdného řádku. Načtený požadavek se předá metodě *processHeader()* objektu třídy *RequestProcessor*. Tato metoda vrací hodnotu typu *int*, udávající, jak

analýza požadavku dopadla. V případě, že je navrácena hodnota -1, šlo v požadavku o metodu POST a je potřeba načíst ještě tělo požadavku. V případě, že je navrácena hodnota 0, proběhla analýza požadavku v pořádku, je-li navráceno číslo vyšší jako 0, jedná se o kód chyby protokolu HTTP. Po analýze požadavku, je možné jednotlivé atributy tohoto požadavku (verzi HTTP protokolu, použitou metodu, žádaný URI, délka těla u metody POST, …) získat pomocí příslušných metod s prefixem *get* (např. *getMethod()*).

V případě, že byl navrácen chybový kód, je předán metodě *sendErrorReply()*, která vytvoří hlavičku pro HTTP odpověď s požadovaným chybovým kódem a jako tělo této odpovědi načte soubor s odpovídajícím HTML kódem. V případě, že nedošlo k žádné chybě, jsou atributy požadavku předány metodě *sendOKReply()*. Tato metoda nejprve dekóduje z předaného URL cestu (např. */groups.html*) a vlastní dotaz (např. *group=pokus.net*). Požadavek na soubory s obrázky, nebo se souborem kaskádového stylu, je obsloužen prostým načtením daného souboru z disku a odesláním v podobě HTTP odpovědi. Při odesílání HTTP odpovědi se volá metoda *composeHttpHeader()*, které se předá délka těla, jenž bude připojeno k odpovědi. Tato metoda vygeneruje kompletní HTTP hlavičku tak, aby byla v souladu s protokolem HTTP/1.1.

Dále jsou obslouženy požadavky na následující umístění: / (*root*), v tomto případě se z disku načte soubor *groups.html*. Tento soubor bude sloužit k zadání *wildmat* řetězce, pro výběr skupin k zobrazení. Po načtení souboru se v něm pomocí regulárního výrazu nahradí řetězec *<?FILTER?>* za * (Což je výchozí *wildmat* řetězec, jehož použití způsobí výpis všech skupin na serveru) a řetězec *<?CHARSET?>* za název znakové sady získaný z konfiguračního souboru. Znaková sada je obdobným způsobem nastavována pro všechny generované HTML stránky. Na zobrazené stránce se uživateli tedy zobrazí vstupní pole pro omezení výpisu seznamu diskusních skupin a tlačítko *Send*. Kliknutí na tlačítko *Send* má za následek zaslání požadavku (metoda GET) na */groups.html?filter=**. Pokud funkce *sendOKReply* detekuje tento požadavek, vytvoří instanci objektu *NNTPClient()* a předá mu údaje nutné pro zabezpečené nebo nezabezpečené připojení ke vzdálenému NNTP serveru. Bohužel Java neumožňuje žádným způsobem zjistit, zda-li je navázané připojení stále aktivní, proto je nutné před každým zasláním smysluplného příkazu NNTP serveru zaslat takzvaný *HEARTBEAT* příkaz, kterým se zjistí, zda je připojení aktivní. V případě, že je, server odpoví, že daný příkaz nezná. V případě, že připojení aktivní není, je nutné prostřednictví metody *quit()* objektu *nntpClient* spojení ukončit a vytvořit novou instanci tohoto objektu. Tato technika je zde z toho důvodu, že protokol HTTP1.1 udržuje perzistentní spojení, během kterého může relace s NNTP serverem vypršet, takhle je relace znovu navázána a uživatel sedící u webového prohlížeče nic nezpozoruje. V případě, že přijde požadavek na seznam skupin, je předán metodě *listActive(String pattern*) objektu *nntpClient*, která zašle na NNTP server příkaz *LIST ACTIVE pattern* a výsledek tohoto příkazu zformátuje jako HTML kód, kdy každý název skupiny je na samostatném řádku a tvoří odkaz na umístění */group.html?group=nazev_skupiny*. Status odpovědi NNTP serveru je návratovou hodnotou funkce *listActive* a v případě, že signalizuje úspěšné provedení, může být výše uvedený HTML kód z NNTP

klienta vyzvednut prostřednictvím metody *getOutput()*. Následně je z disku načtena opět stránka *groups.html*, do které je na místo řetězce *<?NEWSGROUPS/?>* vložen výše uvedený html kód. A na místo řetězce *<?FILTER?>* je vložena hodnota *pattern* použitá v tomto požadavku. V případě, že NNTP server navrátil nějakou chybu, je tato získána voláním metody *getOutput()* a vložena do stránky na místo seznamu skupin. V případě, že metoda *listActive* vrátí hodnotu 480, signalizující, že server vyžaduje autentizaci, zkusí se nejprve vyzvednou z doručeného požadavku autentizační údaje. Tyto údaje jsou obsaženy v hlavičkovém řádku *Authorization*, který má tvar *Authorization: Basic username:password* , řetězec *username:password* je zakódovaný pomocí Base64. Pokud jsou tyto údaje v požadavku obsaženy, provede se prostřednictvím NNTP klienta jednoduchá autentizace zasláním příkazů *AUTHINFO USER username* a *AUTHINFO PASS password*. V případě, že je autentizace úspěšná, volá se metoda *listActive()* znovu. Pokud nejsou autentizační údaje v požadavku obsaženy, navrácena je HHTP odpověď s kódem *401 Unauthorized* a hlavičkou *WWW-Authenticate: Basic realm="ProNNTP"*. Uživateli webový klient zobrazí dialog s přihlašovacím formulářem, kde musí zadat autentizační údaje a webový klient zopakuje požadavek. Tento postup zpracování je opakován u všech následujících požadavků.

Požadavek na získání přehledu článků volá metodu *nntpClient.getOverview(String group)*, která vytvoří HTML kód se seznamem článků v dané diskusní skupině seřazeným podle odpovědní hierarchie, kdy u každé zprávy je její předmět odkazem na *aritcles.html?group=nazev_skupiny&article=cislo_clanku*. Seznamem článků se nahradí výraz *<?NEWSGROUPLIST?>* ve stránce *articles.html*. Tato stránka se odešle klientovi. Kliknutím na odkaz v předmětu článku se vyšle výše uvedený požadavek.

Takovýto požadavek má za následek vyvolání metody *nntpClient.getArticle(group,artNumber)*. Tato metoda načte z NNTP serveru článek a vytvoří instanci třídy *ARTICLE*, která navíc kromě vlastního textu článku obsahuje i odkazy na předchozí a následující článek v diskusní skupině. Tyto odkazy později slouží pro implementaci navigačních tlačítek *prev* a *next* v liště HTML stránky, kdy se v načtené stránce *article.html* nahradí výrazy *<?PREV?>* a *<?NEXT?>* právě těmito odkazy. Dále se nahradí výraz *<?FROM?>* adresou odesílatele, výraz *<?SUBJECT?>* předmětem článku, výraz *<?DATE?>* datem vytvoření článku a výraz *<?BODY?>* vlastním tělem článku.

V okamžiku připojení k NNTP serveru je zjištěno, zda tento server umožňuje zasílání nových článků. V případě, že tomu tak je, je v navigační liště na stránce pro prohlížení skupiny zobrazeno tlačítko *New Article*, nebo na stránce pro prohlížení článků zobrazeno tlačítko *Reply*. Kliknutí na jedno z výše uvedených tlačítek má za následek vygenerování požadavku na *send.html?group=nazev_skupiny&article=cislo_clanku*, který vede k zobrazení formuláře pro zaslání nového článku do diskusní skupiny. V případě, že *cislo_clanku* je 0, neodpovídá se na žádný předešlý článek a je zobrazen pouze prázdný formulář, v případě, že je číslo článku jinačí než nula, odpovídá se na nějaký konkrétní článek a je volána metoda *getRefAndSubject()*, kterou se získá předmět onoho článku předcházený prefixem *Re:* a dále reference (tj. posloupnost Message-ID, tak jak bylo na článek

odpovídáno). Tyto údaje jsou dále vloženy do skrytých (*HIDDEN*) polí formuláře pro odeslání článku ve stránce *send.html* a jsou zaslány i s textem tohoto článku při kliknutí na tlačítko *Send()*. Důležité je ještě zdůraznit, že jakýkoliv text získaný z NNTP serveru, je nutné před vložením do HTML stránky převést na HTML kód, tj. nahradit značky < > ‘’ \ HTML kódem, převést konec řádku na tag *</br>* a každé dvě mezery nahradit sekvencí *mezera *; Tlačítko *Sendt* na formuláři s novým článkem jako jediné generuje metodu POST , URL této metody je stránka *send.html* a tělo požadavku tvoří sekvence *from=[text] &subject=[text] &body=[text] &references=[text]& newsgroup=[text]*. Tento řetězec se předá konstruktoru třídy *ArticleBuild*, který zadaná data zkontroluje a vytvoří z něj článek, který je ve formátu vhodném k zaslání NNTP serveru. Tento článek lze získat z objektu třídy *ArticleBuild* metodou *getArticle()* jako textový řetězec v případě, že metoda *getErrorStatus()* stejného objektu nenavrátí žádnou chybu. Nejprve je zavolána metoda *nntpClient.post()*, která zažádá o zaslání článku, a v případě, že je žádost úspěšně vyřízena je metodou *nnptClient.post2(String clanek)* odeslán článek na diskusní server. Dále je načtena stránka *sent.html*, do které je na místo řetězce *<?BODY/?>* vložena odpověď NNTP serveru na zaslání, popřípadě, vložena chybová zpráva, kterou vygeneroval objekt třídy *ArticleBuild*. Tato stránka je zaslána uživateli. Spojení s webovým klientem je ukončeno v případě, že jde o verzi protokolu HTTP 1.0 nebo je to požadováno hlavičkou *Connection: close* v požadavku protokolu HTTP 1.1.

Popis třídy NNTPClient.java

Jak již bylo uvedeno výše tato třída představuje implementaci *nntpClienta*, který zasílá dotazy serveru a na základě získaných odpovědí generuje HTML kód, který ukládá do proměnné *private String output*, která může být získána metodou *getOutput()* této třídy. V případě, že došlo k chybě, je chybová zpráva vložena do proměnné *output* namísto HTML výstupu. Číselný status odpovědi z NNTP serveru je přímo navracen jednotlivými metodami třídy *NNTPClient*. Na základě tohoto čísla lze rozlišit, zda došlo k chybě a v proměnné output je chybová zpráva nebo požadovaný HTML kód.

Typické zpracování víceřádkové odpovědi na NNTP příkaz probíhá následujícím způsobem. Ze socketu se načte první řádek odpovědi, podle kterého se určí, zda došlo nebo nedošlo k chybě a zda budou následovat další řádky odpovědi. V případě, že budou následovat, čtou se řádky do té doby, než je načten prázdný řádek s tečkou (konec víceřádkové odpovědi). Každý řádek se požadovaným způsobem zpracuje a výsledek se transformuje na HTML kód, který je uložen do již zmiňované proměnné *output*.

V případě volání metody *getOverview(název skupiny)* je využita instance třídy *Overview*, do které se pro každý článek uloží informace o něm získané příkazem *OVER* nebo *XOVER* z NNTP serveru. Jedná se o hodnoty hlaviček *From, Subject, Date, Newsgroups*, *References* a *Message-ID*. Třída *Overview* navíc obsahuje metody, jež usnadní seskládání a odsazení řádků s popisem jednotlivých článků za sebe tak, aby odpovídaly odpovědní hierarchii mezi články, kdy se využívá

především obsahu řádků *Message-ID* a *References* u jednotlivých článků. Hlavičkový řádek *References* obsahuje posloupnost <*Message-ID*> tak, jak je článek řazen v odpovědní hierarchii. Pro každý článek se uloží v jaké úrovni hierarchii je (podle počtu *Message-ID* v hlavičce *References*). Řazení probíhá následujícím způsobem, ze serveru se načte popis nejstaršího článku a uloží se do objektu *Vector* (dynamické pole), vezme se další článek a postupně porovnává první Message-ID z hlavičky *References* s Message-ID článků uložených v poli *Vector*, ale jen s těmi, které mají stejné zanoření v odpovědní hierarchii jako je aktuálně zkoumané zanoření článku. Pokud takový článek nenalezne, vloží zkoumaný článek na konec pole. V případě, že takový článek nalezne, vezme další Message-ID z hlavičky *References*, zvýší se aktuální odpovědní úroveň a porovnává je s následujícími Message-ID článků v poli se stejnou úrovní zanoření, dokud takový nenajde. Pokud najde, bere další Message-ID z řádku *References* a pokračuje průchodem a porovnáváním. Pokud při průchodu polem narazí na článek, který je v odpovědní hierarchii níže, vloží se prohledávaný článek na aktuální pozici v poli. Pokud není splněna ani jedna výše uvedená podmínka, vloží se článek až na konec pole. Nyní jsou v poli uloženy seřazené články. Z jednotlivých objektů tohoto pole, které reprezentují údaje o daných článcích, se vytvoří popisky článků v podobě HTML řádků. Tyto řádky jsou již na správném místě, ale pomocí tagu <*div*> pro ně musí být specifikováno odsazení od kraje stránky tak, aby toto odsazení odpovídalo úrovni v odpovědní hierarchii.

Metoda *getArticle(String group, long artNumber)* nevrací přímo HTML kód, ale instanci objektu třídy *Article*, ze které se příslušný text dostane specializovanými metodami této třídy. Kterými jsou např. *getArticle()* pro získání těla článku, *getSubject()* pro získání hodnoty předmětu článku, atd.

Popis třídy ArticleBuild

Jak již bylo popsáno výše, tato třída slouží k převodu článku a příslušných atributů tohoto článku z formátu daného tělem metody POST na článek vyhovující standardu RFC 3977. Úkolem této třídy je především otestovat správnost jednotlivých důležitých hlavičkových řádků jako je *From*, *Subject*, *References*, *Newsgroups* a doplnit řádek *Date* s aktuálním datem ve správném formátu. U řádků *From* a *Subject* je potřeba překódovat jejich části obsahující ne ASCII znaky do kódování a formátu předepisovaného standardem RFC 2047 [8]. K tomuto účelu slouží metoda *encodeHeaderLine(String line)*. Při zpracování těla je potřeba zdvojit znaky tečka „.“ umístěné na začátcích řádků tak, aby nekolidovali s koncem článku.

Popis třídy Article

Úkolem této třídy je z hlavičky a článku získaného dotazem *ARTICLE* z NNTP serveru vypreparovat prostřednictvím regulárních výrazů hodnoty pro jednotlivé hlavičkové řádky a zpřístupnit je prostřednictvím dotazovacích metod (např. *getSubject()* pro získání hodnoty předmětu článku). Dále

tento objekt ruší zdvojení znaku tečka „.“ na začátcích řádků. Pro získání záhlaví *Subject* a *From* v čitelném stavu je potřeba použít metod třídy Mime.

Popis třídy Mime

Tato třída poskytuje veřejnou metodu *String decodeHeaderLine(String line)*. Pomocí, které se dekódují hodnoty hlaviček *Subject* a *From*, které jsou ve tvaru definovaném standardem RFC2047 [8] a mají následující tvar:

ASCIITEXT=?KODOVANI?ZNAKOVA_SADA?ZAKODOVANY_TEXT?=ASCIITEXT. Kde *ASCIITEXT* je text složený pouze z ASCII znaků, *KODOVANI* je dáno buď písmenem *Q* nebo *B*, kdy *Q* značí kódování podobné kódování Quoted printable a *B* značí kódování Base64, *ZNAKOVA_SADA* je řetězec určující znakovou sadu a *ZAKODOVANY_TEXT* je text zakódovaný uvedeným kódováním. Třída tedy navíc obsahuje veřejnou metodu *base64Decode()* a *quotedDecode()* pro dekódování řetězců z výše uvedených kódů. Metoda *base64Decode()* je navíc použita i při dekódování řádku *Authorization* u HTTP požadavků s autentizačními údaji.

Popis třídy RequestProcessor

Tato třída převezme HTTP požadavek a provede jeho syntaktickou analýzu s tím, že vypreparuje potřebné atributy, kterými jsou URI požadavku, verze HTTP protokolu, zda je obsažena hlavička *Host* a *Date* (u HTTP 1.1 musí být obsaženy), jaká je hodnota hlavičky *Connection* (pokud je *close*, je vyžadováno uzavření spojení) a dále je zjištěn obsah hlavičky *Authorization*, která obsahuje autentizační údaje. Dalším takovým údajem je délka těla u metody POST. Všechno toto zpracování je provedeno v rámci metody *processHeader()*, která vrací hodnotu *int* udávající, jak zpracování proběhlo. V případě, že byla navrácena hodnota 0 (zpracování úspěšné) nebo -1 (zpracování úspěšné, ale nutno načíst tělo metody POST), mohou být hodnoty jednotlivých atributů získány prostřednictvím funkcí s prefixem *get*, např. *getMethod*(), *getHttpVer*(), *getPostBodyLength*(),...

Implementace konfiguračního rozhraní

Jako konfigurační rozhraní slouží dvě samostatné aplikace v Javě s grafickým uživatelským rozhranním. Jedna je určena pro konfiguraci NNTP rozhraní a druhá pro konfiguraci HTTP rozhraní. Tyto aplikace umožňují nastavit jednotlivé volby pro běh NNTP a HTTP rozhraní.

Konfigurátor NNTP serveru dále nabízí grafické prostředí pro práci s databází, kde je možné vytvářet, modifikovat a mazat diskusní skupiny, uživatelské účty, distribuční vzory a účty vzdálených NNTP serverů pro synchronizaci. Samozřejmostí je možnost vymazávat z diskusních skupin nechtěné

příspěvky. Podrobný popis použití uvedených konfiguračních nástrojů je v příloze této práce. Obě aplikace pracují na stejném principu. Základem je objekt třídy *Configuration*, jež obsahuje všechny potřebné konfigurační údaje pro dané rozhraní. Tento objekt disponuje metodami s prefixem *get*, kterými lze z objektu získat hodnotu pro požadovanou položku, např. *getProxyMode()*, pro získání informace o tom, zda má být použit proxy nebo lokální mód. Dále jsou obsaženy metody s prefixem set, kterými lze do objektu vložit hodnotu určitého konfiguračního údaje např. *setDbPassword()* pro vložení hesla k databázi.

Konfigurační aplikace pracuje následovně. Při kliknutí na tlačítko Save setting se zjistí, zda jsou konfigurační údaje zadané prostřednictvím prvků grafického rozhranní vyhovující (např. jestli hodnota pro číslo portu je opravdu číselná). V případě, že tomu tak je, vloží tyto konfigurační údaje do objektu třídy *Configuration*. Dále vytvoří objekt reprezentující speciální vstupně/výstupní proud pro serializaci objektů a uloží celý objekt do souboru. V případě, že je požadováno restartování služby NNTP nebo HTTP rozhraní tlačítkem *Restart Service*, program pomocí příkazů `NET START` a `NET STOP` zajistí restart uvedených služeb. A programy reprezentující webové a NNTP rozhraní si při novém startu načtou konfigurační objekt ze souboru a nastaví podle něj své parametry běhu. Grafická část konfigurátoru je vystavěna nad knihovnou Swing.

Popis tabulek databáze MySQL

Tabulka distribpats

Tabulka slouží k uložení řádků s údaji pro příkaz LIST DISTRIB.PATS.

Položka	Typ dat	Parametry	Integritní omezení	Popis sloupce
Weight	int(11)	NOT NULL		Váha tohoto záznamu
Pattern	varchar(128)	NOT NULL	UNIQUE KEY	Wildmat řetězec
Value	varchar(256)	NOT NULL		Názvy diskusních skupin

Tabulka includes

Tabulka slouží jako spojovací tabulka mezi diskusní skupinou a do ní náležejícím článkem.

Položka	Typ dat	Parametry	Integritní omezení	Popis sloupce
msg_id	varchar(256)	NOT NULL character set latin1	FOREIGN KEY REFERENCES `news` (`msg_id`)	Message-ID článku
Numer	int(16)	unsigned, NOT NULL		číslo článku v dané diskuzní skupině
grp_id	int(6)	unsigned, NOT NULL	FOREIGN KEY REFERENCES `newsgroups` (`id`)	Identifikátor diskusní skupiny

Tabulka news

Tabulka slouží k uložení jednotlivých diskusních příspěvků.

Položka	Typ dat	Parametry	Integritní omezení	Popis sloupce
msg_id	varchar(256)	NOT NULL character set latin1	PRIMARY KEY	Message-ID článku
overview	text			Uložení overview řetězce popisujícího článek
head	text			Uložení textu hlavičky článku
body	text			Uložení textu těla zprávy
time	timestamp	NOT NULL, default CURRENT_TIMESTAMP		Časové razítko udávající čas vytvoření zprávy

Tabulka newsgroups

Tabulka slouží k uložení jednotlivých diskusních skupin.

Položka	Typ dat	Parametry	Integritní omezení	Popis sloupce
id	int(6)	NOT NULL, unsigned, auto_increment	PRIMARY KEY	Identifikátor skupiny
name	varchar(128)	NOT NULL		Jméno diskusní skupiny
description	varchar(128)	NOT NULlL		Popis diskusní skupiny
time	timestamp	default CURRENT_TIMESTAMP		Čas vytvoření diskusní skupiny
p	tinyint(1)	NOT NULL, default 1		Udává zda, je do skupiny dovoleno zasílat
user	varchar(128)	default NULL		Emailová adresa správce skupiny
last	int(16)	NOT NULL default 0		číslo posledního článku ve skupině

Tabulka users

Tabulka slouží k uložení uživatelských účtů.

Položka	Typ dat	Parametry	Integritní omezení	Popis sloupce
name	varchar(128)	NOT NULL	PRIMARY KEY	přihlašovací jméno
pass	varchar(32)	NOT NULL		hash hesla

Tabulka remote

Tabulka slouží k uložení informací potřebných pro synchronizaci se vzdálenými NNTP servery.

Položka	Typ dat	Parametry	Integritní omezení	Popis sloupce
Id	int(6)	auto_increment	UNIQUE KEY	Id záznamu
address	varchar(256)	NOT NULL		Adresa vzdáleného serveru
port	int(6)	unsigned, NOT NULL		číslo portu vzdáleného serveru
wildmat	varchar(256)	NOT NULL		*Wildmat* pro specifikaci názvů diskusních skupin
time	timestamp	NOT NULL		Čas poslední aktualizace
gmt	tinyint(1)	NOT NULL, default 1		Používá se čas GMT

Java aplikace jako služba Windows NT

V Javě neexistuje žádný nativní mechanismus, kterým by šlo aplikaci naprogramovat přímo jako službu Windows NT. Java však přináší velice silné nástroje pro rychlou a robustní implementaci serverových systémů, u kterých je požadavek na běh v rámci služby NT docela častý. Příkladem takové aplikace může být například aplikační server *Tomcat*. Možným řešením tohoto problému je využití JNI (Java Native Interface) a naprogramování aplikaci v jazyce, který může přímo přistupovat k Win32Api. Tato aplikace bude fungovat jako služba NT, spouštějící JVM (Java Virtual Machine) s požadovanou aplikaci v Javě. Tento postup se ukázal jako možné řešení, ale implementace není nijak jednoduchá a vzniklo několik různých projektů, v rámci nichž se méně či více úspěšná řešení tohoto problému podařila vytvořit. Příkladem takových projektů je např. JavaServiceLauncher nebo Java Service Wrapper.

Pro tuto aplikaci byl zvolen projekt JavaService, jehož domovské stránky jsou na webové adrese: http://javaservice.objectweb.org/js_doc_frame.html a který je volně dostupný v rámci licence GPL. JavaService je software, který spravuje rozhranní mezi JVM a WindowsNT Service Handlerem. Využívá k tomu kód napsaný v C++ a Java JNI mechanismy. Mezi hlavní rysy tohoto programu patří, že není potřeba dělat žádné změny v Java aplikaci, která má běžet jako služba, pracuje na systémech Windows NT4, XP, 2003, služba může být konfigurována pro automatický nebo manuální start, existuje integrace s administrativními nástroji systému Windows (např. s Prohlížečem událostí).

Pro běh implementovaných serverových řešení je potřeba mít nainstalováno prostředí JDK minimálně verze 6. Instalace služby probíhá následovně. Nejprve se soubor JavaService.exe (získaný z projektu JavaService) přejmenuje na jméno požadované služby, takže chceme-li aby se služba jmenovalo ProNNTP, přejmenujeme soubor na ProNNTP.exe. Pro instalaci služby spustíme soubor s následujícími parametry (ve složených závorkách psaných kurzívou jsou vysvětlující komentáře):

```
ProNNTP.exe -install Orion  {určení názvu služby}
{JDK_HOME}\jre\bin\{hotspot|server|classic}\jvm.dll {určení cesty k jvm}
-Djava.class.path={NNTP_HOME} {nastavení proměnné classpath}
-start nntp.NNTPServer {cesta ke třídě s metodou main()}
-out {NNTP_HOME}\log\stdout.log  {soubor pro záznamy ze std. výstupu}
-err {NNTP_HOME}\log\stderr.log {soubor pro záznamy z chybového výstupu}
-current {NNTP_HOME} {určení aktuálního pracovního adresáře}
-depends mysql {název služby, která má být spuštěna před touto službou}
-description "NNTP Server"  {Popis služby zobrazí se v Service Control
Manageru}
```

Po provedení tohoto příkazu se nainstaluje služba ProNNTP, která se bude automaticky spouštět při startu Windows a v rámci které poběží Java aplikace určená parametrem *-start*. Tuto službu lze ovládat prostřednictvím konzole *Služby* systému Windows nebo prostřednictvím příkazů NET.

Závěr

V rámci této práce se podařilo navrhnout a implementovat diskusní NNTP server na základě specifikovaných požadavků. V průběhu práce na projektu došlo k vydání standardu RFC 3977, který definuje novou verzi protokol NNTP. Z tohoto důvodu byl původní návrh vycházející ze staré verze NNTP protokolu upraven a implementován byl server podle nového standardu. Vytvořený systém je navíc vybaven režimem kompatibility tak, aby mohl spolupracovat s existujícími NNTP klienty. Dále bylo vytvořeno univerzální webové rozhraní, které lze využít i pro připojení k jakémukoliv jinému NNTP serveru. Při návrhu a implementaci bylo dbáno také na bezpečnost vytvářeného serverového řešení. Server disponuje zabezpečenými variantami použitých protokolů a při implementaci byl brán ohled na zamezení některým taktikám útoků na systém. Implementace serveru v jazyce Java umožňuje nasazení vytvořené aplikace i na jiných operačních systémech.

Objektově orientovaný návrh a důkladná dekompozice do tříd umožňuje poměrně snadné rozšíření vytvořeného systému. Z hlediska dalšího vývoje by bylo dobré doplnit aplikaci mechanismy specifikovanými v rozšířeních nového protokolu NNTP. Dále se nabízí možnost doplnění síťového systému filtrem příchozích spojení, kterým by se dalo zamezit přístupu popř. zasílání článků z určitých IP adres. Vývoje by mohl také spočívat v doplnění možnosti definovat práva přístupu jednotlivých uživatelů k diskusním skupinám tj. definovat, zda má daný uživatel právo skupinu prohlížet, zasílat do ní, zda ji má mít viditelnou v seznamu skupin, atd. Pro zvýšení výkonnosti systému by bylo vhodné nahradit použitý databázový systém MySQL nějakým výkonnějším systémem, popř. vyvinout databázový systém specializovaný pro data, se kterými pracuje protokol NNTP. Systém stahování článků ze vzdálených serverů by bylo vhodné doplnit mechanismem, který dokáže stáhnout nově vytvořené diskusní příspěvky, aniž by využíval příkazu NEWNEWS. Provedení tohoto příkazu je totiž výpočetně náročné a mnoho administrátorů ho na svých NNTP serverech zakazuje. Komfort administrace systému by mohl být zvýšen doplněním webového rozhraní pro vzdálenou správu.

Možnost nasazení systému v praxi byla ověřena několikatýdenním provozem aplikace v menší organizaci, kde server zprostředkovával přístup k několika často používaným diskusním skupinám.

Literatura

[1] CommSoft Systems Development: *An Introduction to NT Services*. 2000. Dokument dostupný naURL http://www.commsoft.com/services.html (květen 2007).

[2] Feather, C.: *RFC 3977 Network News Transfer Protocol*.. 2006. Dokument dostupný na URL http://www.ietf.org/rfc/rfc3977.txt (květen 2007).

[3] Fread, F., Borenstein, F.: *RFC 2045 Multipurpose Internet Mail Extensions (MIME) Part One: Format of Internet Message Bodies*. 1996. Dokument dostupný na URL http://www.ietf.org/rfc/rfc2045.txt (květen 2007).

[4] Halsall, F.: *Computer Networking and the Internet*. Fifth Edition, Addison Wesley, 2005.

[5] Horton, M.: *RFC 850 Standard for Interchange of USENET Messages*. 1983. Dokument dostupný na URL http://www.faqs.org/rfcs/rfc850.html (květen 2007).

[6] Hruška, T., Burget, R.: *Internetové aplikace (WAP) 1*. Brno, 2006. Dokument dostupný naURL https://www.fit.vutbr.cz/study/courses/WAP/private/oporaWAP1.pdf (květen 2007).

[7] Kantor, B., Lapsley, P.: *RFC 977 Network News Transfer Protocol*.. 1986. Dokument dostupný na URL http://www.ietf.org/rfc/rfc0977.txt (květen 2007).

[8] Loupanec, J.: *NNTP server jako služba pro systémy založené na technologii Windows NT*. Semestrální projekt, Brno, FIT VUT v Brně, 2007.

[9] Moore, K.: *RFC 2047 MIME (Multipurpose Internet Mail Extensions) Part Three: Message Header Extensions for Non-ASCII Text*. 1996. Dokument dostupný na URL http://www.faqs.org/rfcs/rfc2047.html (květen 2007).

[10] Network Working Group: *RFC 2068 Hypertext Transfer Protocol -- HTTP/1.1*. 1999. Dokument dostupný na URL http://www.w3.org/Protocols/rfc2616/rfc2616.html (květen 2007).

[11] Vinocour J., Murchinson K.: *RFC 4643 Network News Transfer Protocol (NNTP) Extension for Authentication.*. 2006. Dokument dostupný na URL http://www.eyrie.org/~eagle/nntp/rfcs/rfc4643.txt (květen 2007).

[12] Wikipedie: *Relační databáze*. Dokument dostupný na URL: http://cs.wikipedia.org/wiki/Rela%C4%8Dn%C3%AD_datab%C3%A1ze (květen 2007).

Seznam příloh

[A] Manuál k vytvořenému programu.

Přílohy

A. Manuál k vytvořenému programu

A.1. Instalace programu

Požadavky na systém

Pro úspěšné spuštění aplikace je potřeba mít nainstalované tyto součásti:
Operační systém: MS Windows (minimální verze NT4)
Databázový systém: MySQL (minimální verze 5.0.47)
Java: The Java SE Development Kit (JDK) (minimální verze 6)

Příprava databáze MySQL

1.) Nainstalujte databázi MySQL.

2.) Přihlaste se k databázi prostřednictvím MySQL konzole. K přihlášení použijte heslo administrátora zadané během instalace.

3.) Příkazem

```
INSERT INTO user VALUES('localhost', 'jméno_uživatele', PASSWORD('heslo'),
'N', 'N', 'N', 'N', 'N', 'N', 'N', 'N', 'N', 'N', 'N', 'N', 'N', 'N');
```

vložte nový uživatelský účet se zvoleným jménem (řetězec `jméno_uživatele`) a heslem (řetězec `heslo`).

3.) Vytvořte nové schéma databáze, ve kterém budou později vytvořeny tabulky potřebné pro ukládání dat NNTP serveru. Schéma databáze vytvoříte příkazem

```
CREATE DATABASE 'název_schematu' /*!40100 DEFAULT CHARACTER SET utf8
*/;
```

4.) Nastavte vytvořenému uživatelskému účtu práva k vytvořenému schématu prostřednictvím příkazu

```
INSERT INTO db
VALUES('localhost','název_schematu','jméno_uživatele','Y','Y','Y','Y
','Y','N','Y','N','N','N');
```

5.)Příkazem `FLUSH PRIVILEGES;` zaveďte nového uživatele.

6.) Příkazem `Exit;` ukončete práci s konzolou MySQL.

Instalace služeb NNTP a HTTP rozhraní

1.) Adresář s programem ProNNTP umístěte na libovolné místo v adresářové struktuře tak, aby cesta k adresáři neobsahovala mezeru. To je důležité, protože jinak nebudou fungovat instalační skripty.

2.) V kořenové složce programu se nachází skript *CompileAll.bat*, jehož spuštěním proveďte kompilaci všech součástí programu.

3.) Klikněte pravým tlačítkem na ikonu souboru *InstallProNNTPservice.bat* a zvolte Upravit. V souboru upravte řádek `set JAVA_HOME=C:\Progra~1\Java\jdk1.6.0_01`, tak aby obsahoval cestu k adresáři s Java JDK ve vašem systému. Změny uložte a takto upravený soubor spusťte. Provede se instalace služby ProNNTP reprezentující NNTP rozhraní a její následné spuštění.

4.) Zopakujte 3. krok i pro soubor *InstallProNNTPwebService.bat*. Tento krok má za následek vytvoření služby ProNNTPweb reprezentující HTTP rozhranní a následné spuštění této služby.

5.) Zkontrolujte soubory nntperrlog a httperrlog v kořenovém adresáři programu ProNNTP, které obsahují případná chybová hlášení spuštěných služeb. Další případné problémy lze zjistit pomocí Prohlížeče událostí systému Windows (Start-Ovládací panely-Nástroje pro správu-Prohlížeč událostí).

Vytváření souboru s klíči pro šifrovanou komunikaci

Aplikace umožňuje všechna použitá spojení zabezpečit šifrováním. K tomu je potřeba, aby byl vytvořen takzvaný klíčový sklad (*keystore*). Tento sklad slouží k uložení soukromých klíčů a certifikátů. Dále slouží také k uložení veřejných klíčů serverů, ke kterým je umožněno klientské šifrované spojení. Výchozí implementace klíčového skladu v Javě reprezentuje tento sklad jako soubor chráněný heslem. Vytvořené serverové řešení nedisponuje žádnými mechanismy k administraci tohoto souboru, proto je nutné tuto administraci provádět manuálně. Možným řešením je využití program

keytool (součást java runtime environment, v adresáři /bin). Popis použití tohoto nástroje pro vytvoření skladu s klíči a jeho následné administrace je popsán na webových stránkách http://java.sun.com/j2se/1.4.2/docs/tooldocs/windows/keytool.html.

A.2. Nastavení programu

Konfigurace NNTP rozhraní

Konfigurátor NNTP rozhraní se spustí skriptem *ProNNTPconfig.bat* z kořenového adresáře programu ProNNTP. Po spuštění se zobrazí okno, v jehož spodní části jsou tři tlačítka. Tlačítko *Save settings* provede uložení jednotlivých nastavených voleb do souboru. Tlačítko *Restart service* slouží k restartování služby ProNNTP a tím pádem k načtení nových konfiguračních údajů.

Hlavní nastavení

Toto nastavení se provádí na záložce *General setting* (Seznam obrázků, obr.1). Editační pole *Port* umožňuje nastavit port, na kterém bude NNTP server naslouchat. Zaškrtnutím políčka *Secure server* je možné specifikovat, že má server akceptovat pouze šifrované spojení. Aby mohl být tento typ spojení navázán musí být ještě specifikován soubor se skladem klíčů (*KeystoreFile*) a heslo (*KeystorePassword*) k tomuto souboru. Ve spodní části okna je možno zvolit, v jakém módu NNTP server poběží. Lokální mód (volba *Local server*) udává, že server bude pracovat s lokální databází, proxy mód (volba *Proxy server*) znamená, že příchozí NNTP příkazy budou přeposílány na jiný NNTP server.

Nastavení lokálního serveru

Záložka *Local Setting* (Seznam obrázků, obr.2) slouží k nastavení atributů týkajících se lokálního módu. První editační pole (*Server name*) umožňuje specifikovat jméno serveru. Přepínačem u položky *Posting*: je možné specifikovat, zda je povoleno na server posílat diskusní příspěvky (volba *Enabled*) nebo není (volba *Disabled*). Dále je možné určit maximální velikost zasílaného článku v kilobajtech (*Max. article size*) a časový interval v milisekundách, po kterém vyprší neaktivní spojení se serverem (*Connection timeout*). V panelu *Server database connection setting* se nastavují jednotlivé parametry potřebné pro připojení k databázovému systému. Jedná se o adresu serveru (*Server*:) s databázovým systémem MySQL, port (položka *Port:*), na kterém tento systém naslouchá, přihlašovací jméno (*Login*), heslo (*Password*) a název schématu databáze (*Sheme*).

Nastavení proxy serveru

V rámci záložky *Proxy setting* (Seznam obrázků, obr.3), lze nastavit, adresu (*Remote server*) a port (*Port*) serveru, na který budou přeposílány NNTP příkazy v případě, že NNTP rozhraní běží v proxy módu. Dále je možné určit, že pro spojení se vzdáleným serverem má být použito zabezpečené spojení (*Secure NNTP Client (SSL)*). V tomto případě je též nutné určit soubor s veřejným klíčem vzdáleného serveru (*Keystore file*) a heslo (*Password*) k tomuto souboru.

Synchronizace

Na záložce *Synchronization* je možné nastavit, zda synchronizaci provádět nebo neprovádět (*Synchronization enabled*). A v případě, že má být synchronizace prováděna, určit interval jednotlivých synchronizací.

Databáze

Záložka *Database* umožňuje spravovat informace uložené v databázi. Informace pro přihlášení k databázi se nastavují prostřednictvím záložky *Local Settings*. (Nastavené údaje musejí být uloženy tlačítkem *Save setting*, aby byly použity)

Databáze - Správa skupin a příspěvků

Použití tlačítka *Newsgroups & news* má za následek zobrazení okna se dvěma seznamy. Seznam vpravo (*Newsgroups*) obsahuje názvy diskusních skupin. Prostřednictvím tlačítka *Add* je možné vyvolat formulář pro přidání nové skupiny. Tlačítkem *Edit* formulář pro editaci označené skupiny. Akce tlačítka *Delete* vymaže označenou diskusní skupinu i se všemi články z databáze. V případě, že je vybrána některá diskusní skupina, v seznamu nalevo (*News*) se obrazí seznam předmětů článků, které do zvolené skupiny náležejí. Výběrem článku a klinutím na tlačítko *Open* je možné si článek prohlédnout, tlačítko *Delete* umožňuje smazání označeného článku z diskusní skupiny.

Databáze - Správa uživatelských účtů

Použití tlačítka *Users* má za následek zobrazení okna se seznamem přihlašovacích jmen (*Users*) v databázi uložených uživatelů. Prostřednictvím tlačítek v tomto okně je opět možné přidávat (*Add*), upravovat (*Edit*) a odstraňovat (*Remove*) uživatelské účty.

Databáze - Správa distribučních vzorů

Okno vyvolané stisknutím tlačítka *Distribution Patterns* disponuje seznam distribučních vzorů pro NNTP příkaz LIST DISTRIB.PATS a umožňuje nové hodnoty přidávat (*Add*), měnit (*Edit*) a odebírat (*Remove*).

Databáze - Správa účtů vzdálených NNTP serverů

Okno pro správu účtů vzdálených NNTP serverů se vyvolá stiskem tlačítka *Remote servers*. Prostřednictvím seznamu (*Remote NNTP Servers*) v tomto okně je možné zadávat (*Add*), měnit (*Edit*) a odebírat (*Remove*) jednotlivé údaje obsahující adresu a port NNTP serverů pro synchronizaci. Pro každý záznam je možné určit řetězec *wildmat* vymezující názvy diskusních skupin, jejichž obsah se má ze serveru specifikovaném daným záznamem získat.

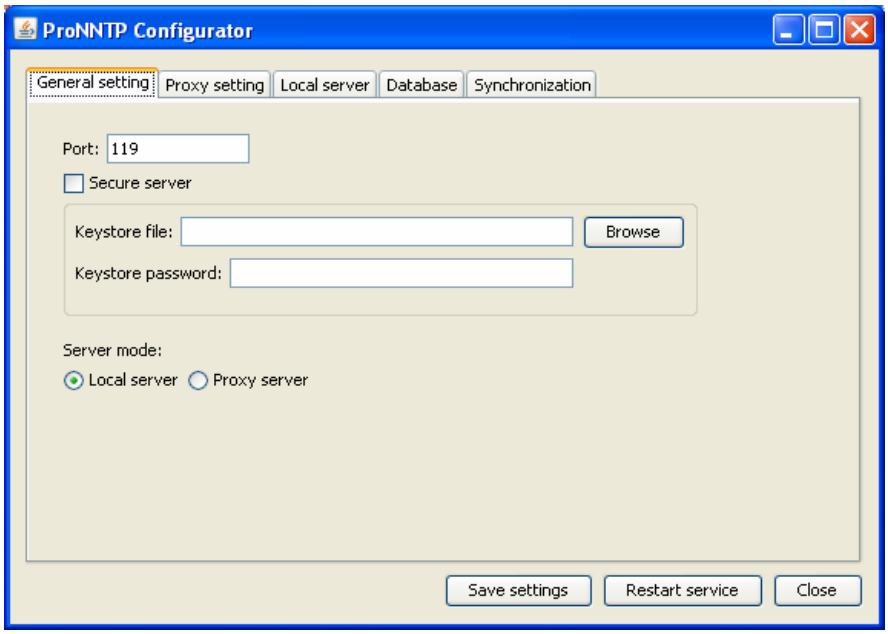
Databáze - Vytvoření tabulek databáze

V případě, že v daném schématu ještě nejsou vytvořeny databázové tabulky pro uložení dat NNTP serveru, stiskem tlačítka *Create DB tables* se tyto tabulky vytvoří.

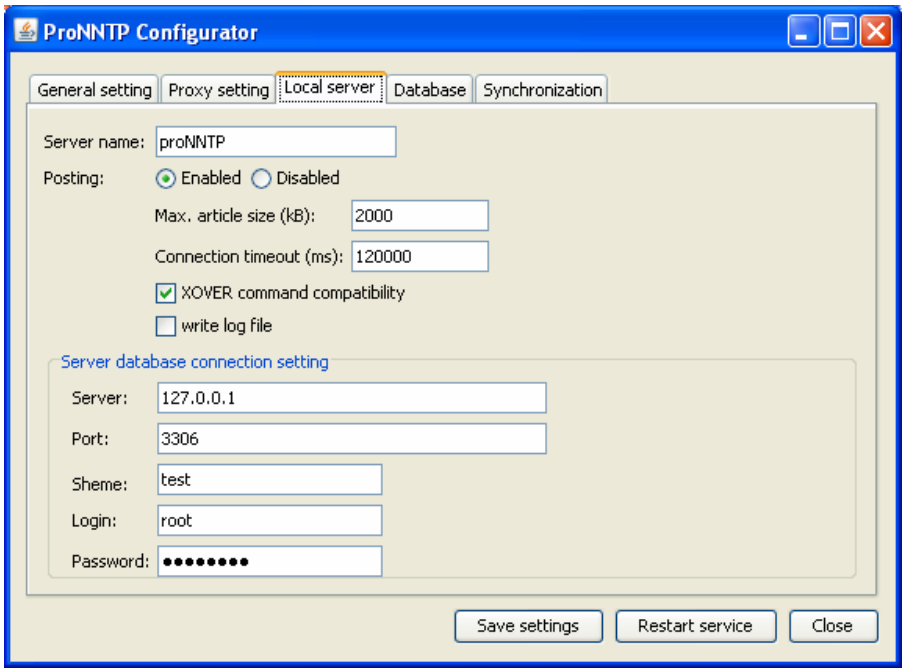
Konfigurace HTTP rozhranní

Program pro konfiguraci HTTP rozhraní se spustí dávkou *ProNNTPwebConfig.bat*, která je umístěna v kořenové adresáři programu ProNNTP. Spuštění této dávky má za následek zobrazení konfiguračního okna (Seznam obrázků, obr.4). Tlačítko *Save* provede uložení jednotlivých nastavených voleb do souboru. Tlačítko *Restart service* slouží k restartování služby ProNNTPweb a tím pádem k načtení uložených konfiguračních údajů.
Konfigurátor umožňuje nastavit port (*Http server port)*, na kterém bude HTTP server naslouchat. Dále je možné zvolit, zda-li má být použito zabezpečené připojení (*Secure HTTP server*). V takovém případě je potřeba nastavit cestu k souboru se skladem privátních klíčů a certifikátů (*KeyStoreFile*) a dále heslo k tomuto souboru (*KeyStorePassword*). Dále je potřeba nastavit adresu (*Remote NNTP server address*) a port (*Remote NNTP server port*) NNTP serveru, ke kterému bude webové rozhraní připojeno. Pro práci s lokálním NNTP serverem je nutné zadat IP adresu 127.0.0.1 a port, na kterém server naslouchá nastavený prostřednictvím konfigurátoru NNTP rozhraní. Pomocí položky *Secure NNTP Server* je možné určit, že ke vzdálenému serveru se má webové rozhraní připojovat zabezpečeným spojením. Pokud je tato volba aktivní, je potřeba ještě určit soubor (*KeyStoreFile*) obsahující veřejný klíč serveru, ke kterému je požadováno připojení a heslo (*KeyStorePassword*) k tomuto souboru. Pomocí editačního pole *Charset* se nastavuje znaková sada html stránek zobrazovaných NNTP rozhraním.

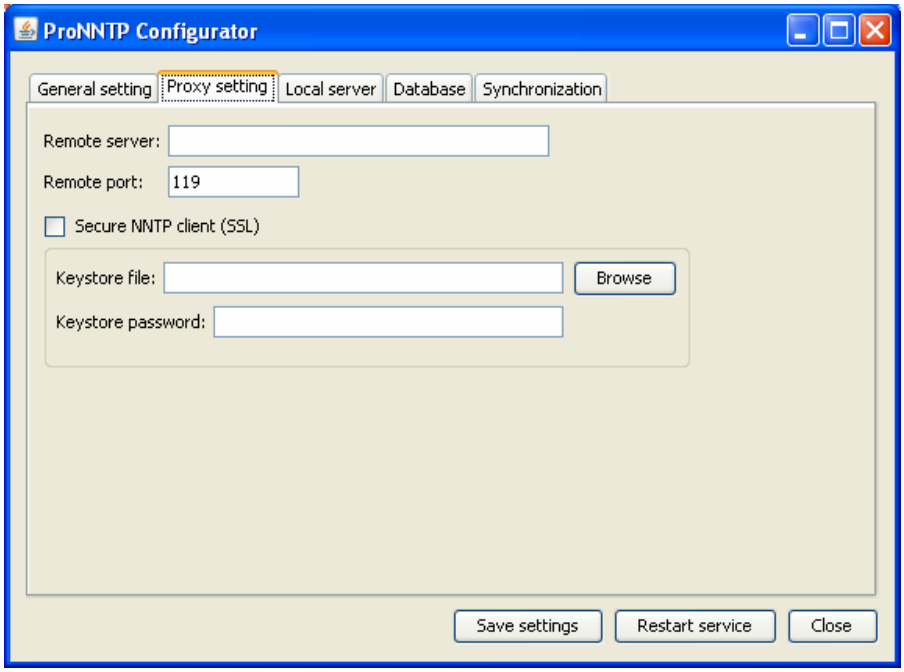
Seznam obrázků



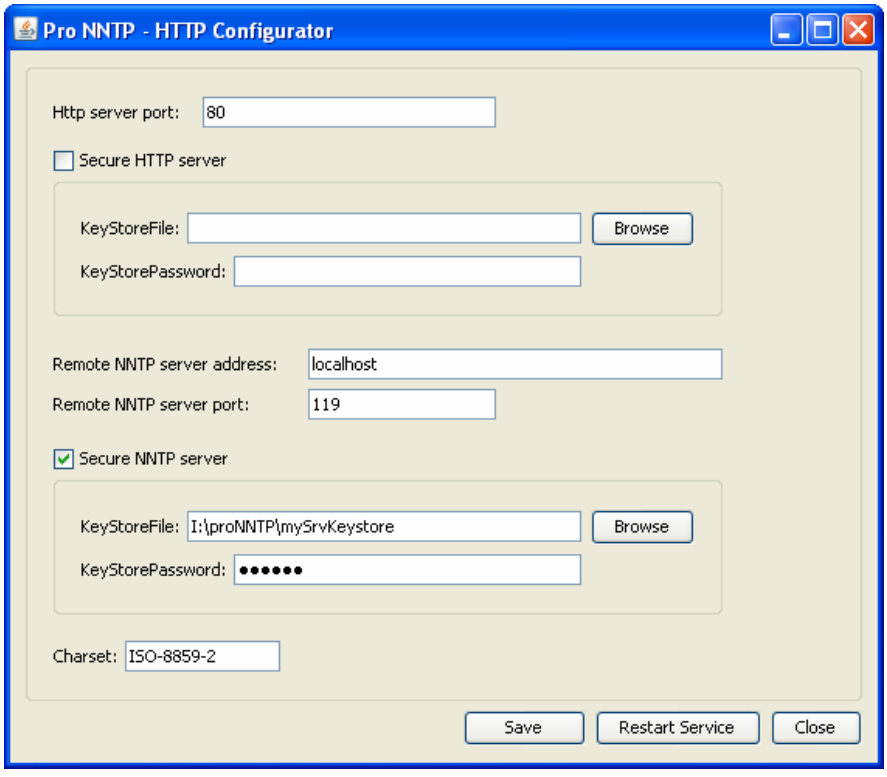
Obr. 1: Záložka General setting



Obr. 2: Záložka Local server



Obr. 3: Záložka Proxy setting



Obr. 4: HTTP Configurator