SMTP & MIME

Jan Chochol

31. prosince 2007

Obsah

1 SMTP

1.1 Úvod

SMTP, nebo-li Simple Mail Transfer Protocol, je internetový protokol určený pro přenos elektronické pošty (e-mailů). Jedná se přímé propojení odesílatele a příjemce.

1.1.1 Historie

Jedná se o jeden z nejstarších protokulů pro přenos e-mailů. Původní verze (RFC 821) byla vydána v roce 1982. Novjejší verze (RFC 2821) byly poté vydána v roce 2001.

1.2 Přenosový kanál

Implementace SMTP nevyžaduje žádný speciální přenosový kanál (tj. nezávisí na něm). Jeiné požadavky na přenosový protokol jsou: spolehlivost, dodržování pořadí odeslaných a přijatých dat. Dnes se používá výhradně TCP standartně na portu 25. Mezi další možnosti přenosu dat patří NCP, NITS, X.25.

1.3 SMTP model

Základní model SMTP je vztah klienta a serveru. Klient vytváří dvoubodové spojení mezi sebou a serverem.
Server může být finálním cílem, nebo také jen smerovač (relay) na další servery. Jinými slovy výměna e-mailu může být buď jedno spojení mezi klientem a serverem, nebo série přeskoků mezi nepřímímy servery.
Po navázání spojení mezi klientem a serverem, klient inicializuje e-mailovou transakci. Takové transakce se skládá z posloupnosti příkazů upřesňujících originátora, příjemce a obsah samotné zprávy. Je-li zpráva určená více klientům, SMTP doporučuje posílání jen jedné kopie zprávy, až do posledního nepřímého serveru, kde již je nutné rozesílat zpávy na jiné servery.
Server odpovídá na každý příkaz odpovědí. Odpověď může být přijetí, požadavek na další příkazy, nebo částečnou či stálou chybu.
Po odeslání jedné zprávy je možné buď pokračovat s novou zprávou, nebo ukončit spojení.
Pokud existuje komunikační kanál mezi originátorem a příjemncem, poté je doporučeno příme spojení. Neexistuje-li takovýto kanál, použije se průchod přes nepřímé servery - v tomto případě se nepřímé servery zjišťují z MX záznamů DNS.
SMTP komunikace využívá transakční model - po navázání spojení klient inicializuje emailovou transakci. Tato transakce se skládá z bloků příkaz

- odpověď. Vytvořený kanál po konci transakce lze znovu využítpro další transakce.

1.4 Příkazy

Jak bylo řečeno SMTP transakce se skládá z bloků příkaz - odpověď. Příkazy jsou jednoduché textové řetězce s možnými parametry. Původní příkazy jsou popsány v RFC 821, následně rozšířeny RFC 2821. Při rozšiřování byl kladen důraz na kompatibilitu, tzn. aby mohla bátprovedena transakce mezi účastníky s libovolnou podporovanou verzí.
Příkazy lze rozdělit do několika kategorií:

1.4.1 Opening and Closing

Do této kategorie spadají příkazy zodpovědné za otevírání a ukončování transakcí. Jednotlivé příkzi jsou následující:

- **HELO**
  Příkazem `HELO` se inicilalizuje emailová transakce. Po obržení tohoto příkazu má sever povinost vymazat všechny stavové informace a vrátit se do stavu na začátku transakce.
  Syntaxe příkazu je následující: `HELO <SP> <domain> <CRLF>`. Položka `domain` má za úkol identifikovat klienta.
  Pro oznámení, že klient podporuje novější verzi SMTP se použije varianta příkazu `EHLO`, která má stejnou syntaxi. Obdržíli server příkaz `EHLO` tak odpoví seznamem rozšíení, které podporuje.

- **QUIT**
  Tímto příkazem se končí aktuální transakce. Jeho syntaxe je `QUIT <CRLF>`

1.4.2 Mail

Do této kategorie spadají příkazy zodpovědné za identifikaci odesílatele/příjemce a samotný přenos dat. Jednotlivé příkzi jsou následující:

- **MAIL**
  Příkaz `MAIL` je první z této skupiny, který je použit po začátku emailové transakce. Jeho úkolem je identifikovat odesílatele zprávy. Server poté provede kontrolu, zda odesílatel může posílat zprávy.
  Syntaxe příkazu je následující: `MAIL <SP> FROM: <reverse-path> <CRLF>`. Položka `reverse-path` identifikuje odesílatele.

- **RCPT**
  Příkaz `RCPT` identifukuje jednoh příjemce. Je-li zpráva určena pro více

příjemců, poté ju nutné tento příkaz vícekrát použít. Po obdržení příkazu server kontrluje zda je povolené posílat zpravy od odesílatele příjemci. Jeho syntaxe je `RCPT <SP> TO: <forward-path> <CRLF>`, kde `foward-path` je identifikátor příjemce.

- **DATA**
  Tento příkaz informuje o úmyslu posílat data. Data se posílají jako 7bitové znaky (norma nedoporučuje posílat kontrolní znaky jiné než SP, HT, CR, LF). Sekvence data je okončena sekvencí `<CRLF> . <CRLF>` - tato sekvence se nesmí v datech nikde nacházet (toto je dosaženo tím, že pokud data obsahují `<CRLF> .`, tak je tato sekvence nahrazena `<CRLF> . .`).
  Syntaxe tohoto příkazu je `DATA <CRLF>`

1.4.3 Verifying and Expanding

Do této kategorie spadají příkazy používané k ověřování dostupnosti příjemců. Jednotlivé příkzi jsou následující:

- **VRFY**
  Tento příkaz slouží k žádosti o kontrolu existence uživatele. Jeho syntaxe je `VRFY <mailbox> <CRLF>`, kde `mailbox` je adresa uživatele na kterou se ptáme.

- **EXPN**
  Tento příkaz slouží k žádosti o expanzi mailing-listu. Jeho syntaxe je `EXPN <mailbox> <CRLF>`, kde `mailbox` je mailing-list na který se ptáme.

1.4.4 Send

Do této kategorie spadají příkazy zodpovědné okamžitý přenos dat. Jednotlivé příkzi jsou následující:

- **SEND**
  Tímto příkazem žádáme o okamžité doručení zpravý na terminál příjemce. Nelze-li okamžitě zprávu doručit, poté server musí nahlásit chybu. Jedná se alternativu příkazu `MAIL`.
  Syntaxe je `SEND <SP> FROM: <reverse-path> <CRLF>`.

- **SOML**
  Send or Mail - tímto příkazem žádáme o okamžité doručení zpravý na terminál příjemce. Nelze-li okamžitě zprávu doručit, poté se zpráva uloží do mailboxu. Jedná se alternativu příkazu `MAIL`.
  Syntaxe je `SOML <SP> FROM: <reverse-path> <CRLF>`.

- **SAML**
  Send and Mail - tímto příkazem se zpráva uloží do mailboxu a poté se server snaží o okamžité doručení zpravý na terminál příjemce. Jedná se alternativu příkazu `MAIL`.
  Syntaxe je `SAML <SP> FROM: <reverse-path> <CRLF>`.

1.4.5 Ostatní

Do této kategorie spadají různé další příkazy, jež nejdou zařadit do žádné z vyšších kategorií. Jednotlivé příkzi jsou následující:

- **HELP**
  Tímto příkazem žádáme o nápovědu/pomoc od serveru. Jeho syntaxe je `HELP [ <SP> <string> ] <CRLF>`, kde `string` je nepovinná položka značící s čím chceme poradit.

- **NOOP**
  Tento příkaz nemá žádný efekt. Jeho syntaxe je `NOOP [ <SP> <string> ] <CRLF>`, kde `string` je nepovinná položka, která nemá žádný význam.

- **RSET**
  Tento příkaz zruší všechny stavové informace o probíhající transakci (server se dostane do stavu stejného jako hned po příkazu `HELO`). Jeho syntaxe je `RSET <CRLF>`.

1.5 Deprecated

V rámci zjednodušení implementace (odstranění málo implementovaných vlastností) a zvětšení bezpečnosti byly nekteré vlastnosti SMTP v RFC 2821 odstraněny. Jedná se především o tyto:

- Příkaz `TURN`
  Tímto příkazem bylo možno zaměnit roli klienta a serveru. Zrušen byl především z bezpečnostních důvodů (za jisté situace šlo nastavit aby server poslal zprávu příjemci, kterému by ji jinak neposlal).

- Sourcve routing
  Tímto prvkem šlo rozšířit příkaz `RCPT` - byl zde zapsán seznam serverů, přes který by měla zpráva putovat. Toto byl hlavně z důvodů, že né všechny servery byli přímo propojeny a neznali adresy všech možných příjemců. V dnešní době internetu jsou většinou servery přímo propojeny pomocí IP protokolu. Adresy příjemců je možné získat z MX záznamů DNS serverů.
  Servery by dnes měli přijímat příkaz `RCPT` se source routingem, ale měli by ho ignorovat (chovat se jakoby tam nebyl).

- Sending
  Příkazy `SEND, SAML, SOML` dnes již nemají velký význam (nelze říct, co je koncový terminál uživatele) a proto bylo od těchto příkazů opuštěno.

1.6 Odpovědi

Jak již bylo řečeno na každý příkaz by měl server odpovědět. Syntaxe odpovědi je `<X> <Y> <Z> <SP> <text>`. Kdy čísla `XYZ` vyjadřují kód odpovědi a `text` vyjadřuje pomocný text. Odpovědi byla navrženy tak, že klient by měl správně reagovat i když jeho imlemetace bere v úvahu jen číslo `X`. Z toho vyplývá, že hlavní (podstatné) členění odpovědí je podle `X`:

- $1yz$ - Vyjadřuje předběžnou pozitivní odpověď - tzn. předpokládá se, žeještě proběhne nějaká akce, která bude úspěšná.
- $2yz$ - Vyjadřuje kompletní pozitivní odpověď.
- $3yz$ - Vyjadřuje dočasnou pozitivní odpověď - tzn. čaká se jěště na nějakou akci, která může ovlivnit výsledek.
- $4yz$ - Vyjadřuje dočasnou chybu - tzn. že při opakování akce se může výsledek změnit.
- $5yz$ - Vyjadřuje trvalou chybu.

Dále norma specifikuje doporučené rozdělení dle `Y` (jedná se o doporučené oblasti použití):

- $x0z$ - syntaxe
- $x1z$ - informace
- $x2z$ - připojení
- $x5z$ - mail system

Příklady odpovědí můžou být následující:

- 211 System status, or system help reply
- 220 ¡domain¿ Service ready
- 250 Requested mail action okay, completed
- 251 User not local; will forward to ¡forward-path¿
- 354 Start mail input; end with ¡CRLF¿.¡CRLF¿
- 450 Requested mail action not taken: mailbox unavailable (e.g., mailbox busy)

- 452 Requested action not taken: insufficient system storage
- 500 Syntax error, command unrecognized
- 501 Syntax error in parameters or arguments
- 552 Requested mail action aborted: exceeded storage allocation
- 554 Transaction failed (Or, in the case of a connection-opening response, "No SMTP service here")

1.7 Příklady

Zde jsou některé příklady komunikace (jako klient jsem já a server je GMail)

```
220 mx.google.com ESMTP 7si99007nfv
HELO myserver.com
250 mx.google.com at your service
MAIL FROM:ja@myserve.com
555 5.5.2 Syntax error 7si99007nfv
MAIL FROM:<ja@myserver.com>
250 2.1.0 OK
RCPT TO:<not@relay.allowed>
550 5.1.1 No such user 7si99007nfv
RCPT TO:<jchochol@gmail.com>
250 2.1.5 OK
DATA
354 Go ahead
Subject: Test

Ahojky!!!!
.
250 2.0.0 OK 1196351466 7si99007nfv
QUIT
221 2.0.0 mx.google.com closing connection 7si99007nfv
```

Další příklad:

```
220 mx.google.com ESMTP f4si477065nfh
EHLO myserver.com
250-mx.google.com at your service, [89.176.9.35]
250-SIZE 28311552
250-8BITMIME
250 ENHANCEDSTATUSCODES
MAIL FROM:<ja@myserver.com>
250 2.1.0 OK
VRFY jchochol@gmail.com
```

```
252 2.1.5 Send some mail, I'll try my best
EXPN jchochol@gmail.com
502 5.5.1 Unimplemented command f4si477065nfh
RSET
250 2.1.0 Flushed f4si477065nfh
HELP
214 2.0.0 http://www.google.com/search?btnI&q=RFC+2821
NOOP
250 2.0.0 OK
SEND
502 5.5.1 Unrecognized command f4si477065nfh
QUIT
221 2.0.0 mx.google.com closing connection f4si477065nfh
```

2 MIME

2.1 Úvod

MIME, nebo-li Multipurpose Internet Mail Exchange Extension, je protokol popisující formát předávaných dat mezi e-mailovými klienty.

2.1.1 Historie

První norma popisující MIME vyšla v roce 1982 pod označením RFC 822. Na rozšiřující se požadavky přenášených dat byla rozšířena v roce 1996 hned 5 RFC: RFC 2045 - RFC 2049. V dnešní době existují další rozšíření, ale ty neruší RFC 204X, ale jen je rozšiřují (př.: RFC 2231).

2.1.2 Protokol

Protokol MIME popisuje metody zakódování a dekódování zpráv. Základní členění je na hlavičky a tělo.

2.2 Základní dělení

2.2.1 Hlavičky

Hlavičky jsou na začátku každé sekce (sekce je buď celá zpráva, nebo tělo nějaké sekce). Existují dva typy hlaviček:

- jednoduché - popisují hlavičky jež informují jen jednou hodnotou - jejich syntaxe je `<header> : <value> <CRLF>`
- složené - používajíse v případě sdělit komplexní informaci jednou hlavičkou - jejich syntaxe je `<header> : <main-value> [; <attribute> = <value>]* <CRLF>`

2.2.2 Tělo

Tělo zprávy obsahuje vlastní data (jejichž struktura a kódování je popsáno hlavičkami). Následuje za hlavičkami oddělené prázdnou řádkou: `<headers> * <CRLF> <body>`. Tělo může být jednodouhé, nebo složené (rozdělené na více sekcí; popsáno níže).

2.3 Významné hlavičky

Významné hlavičky jsou takové které by měli být vždy přítomny a popisují zásadní informace, bez nich nelze dekódovat data:

- `"MIME-Version" ":" 1*DIGIT "." 1*DIGIT`
- `"Content-Type" ":" type "/" subtype *(";" parameter)`

- "Content-Transfer-Encoding" ":" mechanism
- "Content-ID" ":" msg-id
- "Content-Description" ":" *text

2.3.1 MIME-Version

Popisuje verzi použitého standartu. Nutné pro zjištění způsobu dekódování.

2.4 Content-Type

Popisuje data uložená v těle. Tělo muže být jednoduché (pak type je jeden z text, image, audio, video), nebo složené (message, multipart). Další možné typy lze dodefinovat (o toto se stará IANA).

2.5 Složené tělo

Pro složené tělo je duležitý atribut boundary v hlavičce Content-Type, ten říká jakým způsobem jsou jednotlivé části od sebe oděleny. Poté se syntaxe těla dá napsat takto: -- <boundary> <body-part> [<CRLF> -- <boundary> <body-part>]* <CRLF> -- <boundary> --.
Jednotlivé <body-part> pak mají tuto syntaxi MIME-part-headers [CRLF OCTET*] - z tohoto je vidět, že vnitřní struktura opět syntaxticky odpovídá MIME - z toho vyplívá možnost dalšího vnitřního strukturování dat. V tomto případě je OCTET jakýkoliv byte, musí však být dodrženo, že obsah neobsahuje řetězec <boundary>, jinak by nešlo jednoznačně poznat okraje částí.

2.6 Content-Transfer-Encoding

Např z důvodů nasazení toho protokolu u SMTP, nelze někdy přenášet nějaké znaky (např. u SMTP byteys nejvyšším bitem 1). Proto byl uveden mechanismus, jak jednotlivá data zákodovat pomocí omezené abecedy. Toto je poté pospsán v hlavičce Content-Transfer-Encoding.

2.6.1 7bit, 8bit, binary

Tyto kódování neprovásí žánou transformaci (obsahuje nezměněné znaky). Jenom informuje o omezení použité abecedy. Tyto typy kódování jsou jediné povolené pro složená těla.

2.6.2 quoted-printable

Toto kódování je vhodné pro data obshující převážně tisknutelné ASCII znaky. Transformaci lze provést jedním z následujících způsobů (na algoritmu komprese jak zvolit nejvhodnější způsob):

- reprezentace jakéhokoliv oktetu jako `= <2 characters octet hex code>`
- literály (znaky 33-60, 62-126) nezměněné
- HT a SP nezměněné (jen nejsou-li na konci)
- CRLF reprezantované jako line-break
- soft line breaks (jalikož je omezena maximální délka řádku se zákodovanou informaci je nutné nějak říci, že tento konec řádku nepatří do dat) - konec řádku je `=`

2.6.3 base64

Toto kódování je vhodné pro binární data (obshující málo tisknutelných ASCII znaků). Velikost počtu znaků po zakódování naroste o 33%. Kódování spočívá v rozdělení textu na skupinu po 24bitech. Tato 24-bitice ja pak reprezentována 4 znaky z base64 abecedy. Nelze-li utvoří 24-bitici (např. z důvodu konce textu) použije se padding znak místo 1 (nebo 2) base64 znaků.

2.7 Non-ASCII text

S nástupem národních abeced vznikla potřeba reprezentovat hodnoty některých hlaviček i jinými znaky než je 128 ASCII znaků. Pro toto se používá mechanismus zvaný Encoded-word. Encoded word má syntaxi `"=?" charset "?" encoding "?" encoded-text "?="`, kde jednotlivé části znamenají:

- `charset` indentifikuje znakovou sadu
- `encoding` popisuji reprezentaci dat, a může být jedno z:
  - B - base64
  - Q - podobné quoted-printable (až na tyto vyjímky: SP lze reprezentovat jako jako _, tisknutelné znaky nezměněné až na tyto: =, ?, _)

Tento způsob nelze použít a hodnotách atributů složených hlaviček (pr ty lze využít např.: mechanismus popsaný v RFC 2231, i když vetšina e-mailových klientů používá tento mechanismus v rozporu s normou - např.: Gmail, Seznam, Centrum, Outlook, Outlook Express). Jako příklad zakódovaného řetězce může být:

- `=?iso-8859-8?b?7eXs+SDv4SDp7Oj08A==?=`
- `=?ISO-8859-1?Q?Patrik_F=E4ltstr=F6m?=`

2.8 Příklady

Jako příklad MIME zprávy může být např.:

```
MIME-Version: 1.0
From: Nathaniel Borenstein <nsb@nsb.fv.com>
To: Ned Freed <ned@innosoft.com>
Date: Fri, 07 Oct 1994 16:15:05 -0700 (PDT)
Subject: A multipart example
Content-Type: multipart/mixed; boundary=unique-boundary-1

--unique-boundary-1

... Some text appears here ...
--unique-boundary-1
Content-type: text/plain; charset=US-ASCII

This could have been part of the previous part, but
illustrates explicit versus implicit typing of body
parts.
--unique-boundary-1
Content-Type: multipart/parallel; boundary=unique-boundary-2

--unique-boundary-2
Content-Type: audio/basic
Content-Transfer-Encoding: base64

... base64-encoded 8000 Hz single-channel
    mu-law-format audio data goes here ...
--unique-boundary-2
Content-Type: image/jpeg
Content-Transfer-Encoding: base64

... base64-encoded image data goes here ...
--unique-boundary-2--

--unique-boundary-1
Content-type: text/enriched

This is <bold><italic>enriched.</italic></bold>
<smaller>as defined in RFC 1896</smaller>
--unique-boundary-1
Content-Type: message/rfc822

From: (mailbox in US-ASCII)
```

```
To: (address in US-ASCII)
Subject: (subject in US-ASCII)
Content-Type: Text/plain; charset=ISO-8859-1
Content-Transfer-Encoding: Quoted-printable

... Additional text in ISO-8859-1 goes here ...
--unique-boundary-1--
```

3 Závěr

3.1 Zdroje

- RFC - 821, 822, 2045 - 2049, 2821