分散システム特論 レポート
RFC1939 Post Office Protocol - Version3 (POP3) について

情報工学専攻 M1 天野研 木村健一郎

1999 年 2 月 1 日

1.1　POP について

まず始めに POP について簡単に説明することにする。

インターネット上のある種の小さなノードではメッセージ配送システムを運営するだけの資源が足りないことがある[1]。例としては、商用プロバイダに一時的に PPP を用いて接続する PC が挙げられる。この PC の場合は「接続性 (Connectivity)」という資源が不足していると言える。

しかし、これらのノードでもメッセージ配送システムを使用することは非常に有意義である。そこで、あるサーバでメッセージを一元管理し、クライアントがネットワーク越しにそのメッセージを参照・更新できるような仕組みを提供することを考える。そのためのプロトコルが POP である。

これ以降メッセージ = 電子メールであるとして話を進める。

1.2　POP の基本動作

次に、POP の基本動作について述べる。

まず POP サーバは 110 番ポートを listen し、要求が来ていないかを調べる。クライアントからの要求が来ると、サーバとクライアントはコマンドと応答をやり取りする。これが動作の基本となる。まず、コマンド及び応答の概要を述べる。

1.2.1　コマンド

コマンドは以下のような特徴を持つ。

- キーワードと引数から成る
- 大文字と小文字の区別のない printable な ASCII 文字である
- 全てのコマンドは CRLF で終了
- キーワードと引数及び引数と引数の間は 1 つの空白で区切られる
- キーワードは 3 ~ 4 文字、各引数は 40 文字までである

例えばコマンドの例としては

```
RETR 1
```

というものがある。この場合、RETR がキーワード、1 が引数になる。

[1] ここで言うメッセージとは主に電子メールを指し、その配送システムというのは SMTP(Simple Mail Transfer Protocol) を指す。

1.2.2 応答

応答は次のような特徴を持つ。

- 状態を示す文字列とキーワードから成る
- 状態は Positive と Negative の 2 つ
- Positive を表す文字列は"+OK"、Negative は"-ERR" である
- 応答が複数行に渡る場合、各行は CRLF で終わり、最後の行は".CRLF" で終わる
- もし、途中の行の先頭の文字が"." になる場合は、行頭にもう 1 つ"." を追加する (byte-stuff)
- クライアントは行頭に"." が来たら次の文字を調べる。CRFL なら終了、違うなら"." を 1 つ取り除く

".CRFL" によって複数行に渡る応答の最終行を示すため、応答の途中で". CRLF" が入らないように byte-stuff というものが定められているのが特徴である。

応答の例としては

```
+OK 2 320
```

というものがある。この場合、あるコマンドに対する状態が Positive(コマンドが成功した) であり、その応答が 2 320 という 2 つの数字ということである。

1.3 サーバの状態遷移と動作の概要

POP サーバは 3 つの状態を持つ。そして、その状態に応じてクライアントから受け付けるコマンドや応対、そして動作が異なっている。サーバの状態は以下の 3 つである。

- AUTHORIZATION 状態
- TRANSACTION 状態
- UPDATE 状態

これより、それぞれの状態及びその遷移と動作について述べる。

1.3.1 AUTHORIZATION 状態

この状態ではクライアントの認証 (Authorization) を行う。POP サーバはクライアントからの接続要求が来ると、クライアントに「あいさつ (Greeting message)」を返し、この状態に遷移する。

クライアントとの間で幾つかのコマンドと応答をやり取りし、クライアントの認証を行う。認証に成功するとサーバは"+OK" を返し、失敗すると"-ERR" を返す。認証に失敗した後、再び認証を待つか接続を切るかは実装依存である。

サーバはクライアントの認証成功後、メールスプールの適切な Lock を得ようとする。この Lock が取得できなかったときもサーバは"-ERR" を返す。

以下にこの状態で用いるコマンドを挙げる。

コマンド	引数	機能
USER	ユーザ名	ユーザ名をサーバに伝える。この後 PASS コマンドを使い、認証を行う
PASS	パスワード	パスワードをサーバに伝える
APOP	ユーザ名、MD5 のダイジェストメッセージ	APOP による認証を行う

APOP による認証では、通常のパスワードではなく MD5 のダイジェストメッセージを用いる。これにより、パスワードが平文でネットワークを流れる危険がなくなる。クライアントは接続時にサーバから送られる Greeting Message 中の文字とパスワードを繋いだ文字列を MD5 にかけ、それを送って認証を行う。最初にサーバから送られるメッセージは毎回異ならなければならない。

1.3.2 TRANSACTION 状態

この状態ではサーバのメールスプールの状態を取得したり変更したりする。サーバはスプールの Lock を取得したときにスプール中のメッセージに番号を振る。これ以降サーバはこの番号によってメッセージを管理する。

以下にこの状態で用いるコマンドを挙げる。

コマンド名	引数	機能	応答
STAT	なし	スプールの状態を取得する	メッセージ数、オクテット単位のファイルサイズ
LIST	メッセージ番号	特定のメッセージのサイズを取得する	引数で指定されたメッセージの番号とサイズ。無指定ならば全てのメッセージが対象になる。指定したメッセージがなかったり、削除マークがついていたら Negative を返す
RETR	メッセージ番号	メッセージを取得する	byte-stuff に注意してメッセージの本体を送る。指定したメッセージがなかったり、削除マークがついていたら Negative を返す
DELE	メッセージ番号	メッセージに削除マークをつける	メッセージに削除マークをつけ、Positive を返す。既に削除マークがついていたり、メッセージが無いなら Negative を返す。ここではマークをつけるだけで、実際の削除は次の UPDATE 状態で行われる。
NOOP	なし	なし	Positive を返す
RSET	なし	削除マークを取り除く	全ての削除マークを取り除き、Positive を返す
QUIT	なし	UPDATE 状態に移行する	UPDATE 状態に移行し、“+OK” を返す

サーバは各コマンドの応答において、規定された以上の情報を付加してもかまわないが、RFC1939 では推奨していない。代りに以下のようなオプションコマンドを用意している。

コマンド名	引数	機能	応答
TOP	メッセージ番号、非負の行数 (整数値)	メッセージの最初から数行を取得する	メッセージの本体の最初の n 行を返す
UIDL	メッセージ番号 (省略可)	メッセージ固有の数字を返す	メッセージの unique-id を返す。unique-id はサーバの決定する任意の文字列で、0x21 ~ 0x7e の範囲の文字 1 ~ 70 で構成される

1.3.3 UPDATE 状態

この状態では、TRANSACTION 状態で削除マークを付けられたメッセージを実際に取り除く。消去の成功・不成功に関わらずスプールの Lock を解放し、TCP コネクションを閉じる。

また、TRANSACTION 状態からは QUIT コマンド以外の方法で UPDATE 状態にはならない。Timeout 等で処理を中断するときには UPDATE 状態になってはならない。

1.4 実装例

ここでは、これまでの説明で「実装依存」としてきた部分について実際の実装を見てみる。サーバの実装としては qpopper-2.53 を参照した。qpopper については以下の URL にソースコードが置いてある。

`ftp://ftp.qualcomm.com/eudora/servers/unix/popper/qpopper2.53.tar.Z`

また、クライアントの実装としては fetchhmail を参考にした。しかし、クライアントについては自分で実際に実装してみるのが一番よく分かると言うことで、自作のプログラムに POP でのメール着信通知機能を実装してみた。

`fetchhmail:`
`ftp://ftp.ccil.org/pub/esr/fetchmail/fetchmail-4.6.7.tar.gz`

自作プログラム:
`http://www.db.is.kyushu-u.ac.jp/~kimura/xhisho/index-j.html`

1.4.1 APOP のメッセージ

APOP での認証で使う、Greeting Message に含まれる文字列である。RFC では「毎回異なること」とだけ決められている。ただし、RFC1939 にはその実装例として次のようなものが挙げられている。

<process-ID.clock@hostname>

process-ID	:プロセス ID の 10 進数
clock	:システムクロックの 10 進数
hostname	:サーバのホスト名

qpopper ではこの実装例をそのまま実装している。

1.4.2 それ以外の認証方式

USER 及び PASS コマンドを使った認証と APOP による認証だけが RFC1939 で定められている。しかし、これ以前の POP で使われていた RPOP というものも存在し、qpopper では RPOP での認証もサポートしている。しかし RPOP については RFC1460 で obsolete となっているので使用しないことが望ましい。

1.4.3 UIDL について

TRANSACTION 状態でのオプションコマンド"UIDL" については、RFC では「メッセージ固有の番号を返す」としか規定されていない。

qpopper ではまず新しいメッセージにユニークな ID を振るが、それを覚えておく術がない。そこで、メッセージのヘッダに「X-UIDL:」というフィールドを付加し、ここに UIDL を書き込む。これ以降、qpopper はクライアントからのコマンド" UIDL" に対しては、この X-UIDL に書いてあるユニークな ID を返すようにしている。

1.5 セキュリティについての考察

ここで、POP での認証におけるセキュリティについて各認証方式を比較してみる。

まず普通の認証 (USER 及び PASS コマンドを用いる) については、一番危険なのは「ネットワーク上をパスワードが平文で流れる」という点である。これが危険であることは言うまでもない。

そこで、ネットワーク上を平文で流れないようにしたのが APOP での認証である。これはネットワーク上を流れるのは MD5 でのダイジェストメッセージであり、しかも毎回異なることが保証されているのでその点では安全と言える。

しかし、MD5 の性質上 APOP での認証を行うためにサーバは各ユーザのパスワードを「平文」で保持している必要がある。そのため、サーバがクラックされることを考えるとこの方法も安全とは言いがたい。

RPOP については既に obsolete であることも考えれば使わないことが望ましい。RPOP は認証に rlogin と同様の、Reserved-port を用いた認証を行っている。

そこで、より安全に POP での認証を行う上での方針が幾つか考えられる。

- APOP を使う場合、通常の login パスワードとは異なるパスワードを用いる
- USER 及び PASS コマンドでの認証を行い、通信経路自体を暗号化する
- OTP(One Time Password) などの他の認証方式を使用できるようにする

1 番目は APOP をより安全に使おう、と言う方向である。サーバでのパスワードの保持に気をつけ、さらに破られた場合の被害を最小に食い止めるよう心がける必要がある。

2 番目は IPv6 や ssh で実現できる、通信経路自体の暗号化である。これを用いればメッセージ自体も全て暗号化されるので、非常に安全と言える。ただし、これにはかなりのマシンパワーを要するであろう。

3 番目については、残念ながら現在のところ RFC では述べられていないので実装依存となってしまう。しかし、fetchhmail では OTP での認証もサポートしているようである。これはこれからの展開に期待したい。

これらの中では 1 番安全であるのはおそらく 2 番目の方法であろう。しかし、最終的には「誰を信頼するか?」での trade-off になると考える。つまり、サーバを信頼する (APOP) か、通進路を信頼する (USER 及び PASS) か、ということである。

以上のようなことと実際的な問題をふまえた上で、サイトの運用方針を定めることが大切である。

1.6 まとめと感想

以上に述べてきたように、POP3 とはネットワーク越しにサーバのメールスプールを操作するためのプロトコルである。コマンドがかなり多くあり、サーバの状態遷移などがあるのでやや難しいように見えるが、基本的には単純なプロトコルである。また、クライアントについては最小限の実装ならば非常に簡単に行える。

しかし、POP については長期的にサーバにメッセージを保存しておくような利用形態には向いておらず、その辺りを解決する物として現在 IMAP4 というプロトコルも考案されている。これについては機会があれば調べてみたい。

現在商用プロバイダに一時的に接続して電子メールを使う人が増えており、そのような形態においては POP3 が一番よく用いられている。それ以外にも大学の研究室のように多くのマシンを有する環境などでも用いられている。そう言う意味では実際に接する機会が非常に多いプロトコルであり、今回の発表で POP3 について詳しく知る機会を得たことは大変良かったと思っている。また、MD5 などのセキュリティについても知ることができたので、このプロトコルを題材に選んで良かったと思う。