SMTP Service Extension
for Command Pipelining

Status of this Memo

   This document specifies an Internet standards track protocol for the
   Internet community, and requests discussion and suggestions for
   improvements. Please refer to the current edition of the "Internet
   Official Protocol Standards" (STD 1) for the standardization state
   and status of this protocol. Distribution of this memo is unlimited.

Abstract

   This memo defines an extension to the SMTP service whereby a server
   can indicate the extent of its ability to accept multiple commands in
   a single TCP send operation. Using a single TCP send operation for
   multiple commands can improve SMTP performance significantly.

Introduction

   Although SMTP is widely and robustly deployed, certain extensions may
   nevertheless prove useful. In particular, many parts of the Internet
   make use of high latency network links.

   SMTP's intrinsic one command-one response structure is significantly
   penalized by high latency links, often to the point where the factors
   contributing to overall connection time are dominated by the time
   spent waiting for responses to individual commands (turnaround time).

   In the best of all worlds it would be possible to simply deploy SMTP
   client software that makes use of command pipelining: batching up
   multiple commands into single TCP send operations. Unfortunately, the
   original SMTP specification [1] did not explicitly state that SMTP
   servers must support this. As a result a non-trivial number of
   Internet SMTP servers cannot adequately handle command pipelining.
   Flaws known to exist in deployed servers include:

   (1)   Connection handoff and buffer flushes in the middle of
          the SMTP dialogue. Creation of server processes for
          incoming SMTP connections is a useful, obvious, and
          harmless implementation technique. However, some SMTP
          servers defer process forking and connection handoff

until some intermediate point in the SMTP dialogue.
When this is done material read from the TCP connection
and kept in process buffers can be lost.

(2)   Flushing the TCP input buffer when an SMTP command
fails. SMTP commands often fail but there is no reason
to flush the TCP input buffer when this happens.
Nevertheless, some SMTP servers do this.

(3)   Improper processing and promulgation of SMTP command
failures. For example, some SMTP servers will refuse to
accept a DATA command if the last RCPT TO command
fails, paying no attention to the success or failure of
prior RCPT TO command results. Other servers will
accept a DATA command even when all previous RCPT TO
commands have failed. Although it is possible to
accommodate this sort of behavior in a client that
employs command pipelining, it does complicate the
construction of the client unnecessarily.

This memo uses the mechanism described in [2] to define an extension
to the SMTP service whereby an SMTP server can declare that it is
capable of handling pipelined commands. The SMTP client can then
check for this declaration and use pipelining only when the server
declares itself capable of handling it.

1. Framework for the Command Pipelining Extension

The Command Pipelining extension is defined as follows:

(1)   the name of the SMTP service extension is Pipelining;

(2)   the EHLO keyword value associated with the extension is
PIPELINING;

(3)   no parameter is used with the PIPELINING EHLO keyword;

(4)   no additional parameters are added to either the MAIL
FROM or RCPT TO commands.

(5)   no additional SMTP verbs are defined by this extension;
and,

(6)   the next section specifies how support for the
extension affects the behavior of a server and client
SMTP.

2. The Pipelining Service Extension

   When a client SMTP wishes to employ command pipelining, it first
   issues the EHLO command to the server SMTP. If the server SMTP
   responds with code 250 to the EHLO command, and the response includes
   the EHLO keyword value PIPELINING, then the server SMTP has indicated
   that it can accommodate SMTP command pipelining.

2.1. Client use of pipelining

   Once the client SMTP has confirmed that support exists for the
   pipelining extension, the client SMTP may then elect to transmit
   groups of SMTP commands in batches without waiting for a response to
   each individual command. In particular, the commands RSET, MAIL FROM,
   SEND FROM, SOML FROM, SAML FROM, and RCPT TO can all appear anywhere
   in a pipelined command group. The EHLO, DATA, VRFY, EXPN, TURN,
   QUIT, and NOOP commands can only appear as the last command in a
   group since their success or failure produces a change of state which
   the client SMTP must accommodate. (NOOP is included in this group so
   it can be used as a synchronization point.)

   Additional commands added by other SMTP extensions may only appear as
   the last command in a group unless otherwise specified by the
   extensions that define the commands.

   The actual transfer of message content is explicitly allowed to be
   the first "command" in a group. That is, the RSET/MAIL FROM sequence
   necessary to initiate a new message transaction can be placed in the
   same group as the final transfer of the headers and body of the
   previous message.

   Client SMTP implementations that employ pipelining MUST check ALL
   statuses associated with each command in a group. For example, if
   none of the RCPT TO recipient addresses were accepted the client must
   then check the response to the DATA command -- the client cannot
   assume that the DATA command will be rejected just because none of
   the RCPT TO commands worked. If the DATA command was properly
   rejected the client SMTP can just issue RSET, but if the DATA command
   was accepted the client SMTP should send a single dot.

   Command statuses MUST be coordinated with responses by counting each
   separate response and correlating that count with the number of
   commands known to have been issued. Multiline responses MUST be
   supported. Matching on the basis of either the error code value or
   associated text is expressly forbidden.

   Client SMTP implementations MAY elect to operate in a nonblocking
   fashion, processing server responses immediately upon receipt, even

if there is still data pending transmission from the client's
previous TCP send operation. If nonblocking operation is not
supported, however, client SMTP implementations MUST also check the
TCP window size and make sure that each group of commands fits
entirely within the window. The window size is usually, but not
always, 4K octets. Failure to perform this check can lead to
deadlock conditions.

Clients MUST NOT confuse responses to multiple commands with
multiline responses. Each command requires one or more lines of
response, the last line not containing a dash between the response
code and the response string.

2.2. Server support of pipelining

A server SMTP implementation that offers the pipelining extension:

  (1)   MUST NOT flush or otherwise lose the contents of the
         TCP input buffer under any circumstances whatsoever.

  (2)   SHOULD issue a positive response to the DATA command if
         and only if one or more valid RCPT TO addresses have
         been previously received.

  (3)   MUST NOT, after issuing a positive response to a DATA
         command with no valid recipients and subsequently
         receiving an empty message, send any message whatsoever
         to anybody.

  (4)   SHOULD elect to store responses to grouped RSET, MAIL
         FROM, SEND FROM, SOML FROM, SAML FROM, and RCPT TO
         commands in an internal buffer so they can sent as a
         unit.

  (5)   MUST NOT buffer responses to EHLO, DATA, VRFY, EXPN,
         TURN, QUIT, and NOOP.

  (6)   MUST NOT buffer responses to unrecognized commands.

  (7)   MUST send all pending responses immediately whenever
         the local TCP input buffer is emptied.

  (8)   MUST NOT make assumptions about commands that are yet
         to be received.

  (9)   SHOULD issue response text that indicates, either
         implicitly or explicitly, what command the response
         matches.

The overriding intent of these server requirements is to make it as
easy as possible for servers to conform to these pipelining
extensions.

3. Examples

Consider the following SMTP dialogue that does not use pipelining:

```
S: <wait for open connection>
C: <open connection to server>
S: 220 innosoft.com SMTP service ready
C: HELO dbc.mtview.ca.us
S: 250 innosoft.com
C: MAIL FROM:<mrose@dbc.mtview.ca.us>
S: 250 sender <mrose@dbc.mtview.ca.us> OK
C: RCPT TO:<ned@innosoft.com>
S: 250 recipient <ned@innosoft.com> OK
C: RCPT TO:<dan@innosoft.com>
S: 250 recipient <dan@innosoft.com> OK
C: RCPT TO:<kvc@innosoft.com>
S: 250 recipient <kvc@innosoft.com> OK
C: DATA
S: 354 enter mail, end with line containing only "."
 ...
C: .
S: 250 message sent
C: QUIT
S: 221 goodbye
```

The client waits for a server response a total of 9 times in this
simple example. But if pipelining is employed the following dialogue
is possible:

```
S: <wait for open connection>
C: <open connection to server>
S: 220 innosoft.com SMTP service ready
C: EHLO dbc.mtview.ca.us
S: 250-innosoft.com
S: 250 PIPELINING
C: MAIL FROM:<mrose@dbc.mtview.ca.us>
C: RCPT TO:<ned@innosoft.com>
C: RCPT TO:<dan@innosoft.com>
C: RCPT TO:<kvc@innosoft.com>
C: DATA
S: 250 sender <mrose@dbc.mtview.ca.us> OK
S: 250 recipient <ned@innosoft.com> OK
S: 250 recipient <dan@innosoft.com> OK
S: 250 recipient <kvc@innosoft.com> OK
```

```
S: 354 enter mail, end with line containing only "."
 ...
C: .
C: QUIT
S: 250 message sent
S: 221 goodbye
```

The total number of turnarounds has been reduced from 9 to 4.

The next example illustrates one possible form of behavior when
pipelining is used and all recipients are rejected:

```
S: <wait for open connection>
C: <open connection to server>
S: 220 innosoft.com SMTP service ready
C: EHLO dbc.mtview.ca.us
S: 250-innosoft.com
S: 250 PIPELINING
C: MAIL FROM:<mrose@dbc.mtview.ca.us>
C: RCPT TO:<nsb@thumper.bellcore.com>
C: RCPT TO:<galvin@tis.com>
C: DATA
S: 250 sender <mrose@dbc.mtview.ca.us> OK
S: 550 remote mail to <nsb@thumper.bellore.com> not allowed
S: 550 remote mail to <galvin@tis.com> not allowed
S: 554 no valid recipients given
C: QUIT
S: 221 goodbye
```

The client SMTP waits for the server 4 times here as well. If the
server SMTP does not check for at least one valid recipient prior to
accepting the DATA command, the following dialogue would result:

```
S: <wait for open connection>
C: <open connection to server>
S: 220 innosoft.com SMTP service ready
C: EHLO dbc.mtview.ca.us
S: 250-innosoft.com
S: 250 PIPELINING
C: MAIL FROM:<mrose@dbc.mtview.ca.us>
C: RCPT TO:<nsb@thumper.bellcore.com>
C: RCPT TO:<galvin@tis.com>
C: DATA
S: 250 sender <mrose@dbc.mtview.ca.us> OK
S: 550 remote mail to <nsb@thumper.bellore.com> not allowed
S: 550 remote mail to <galvin@tis.com> not allowed
S: 354 enter mail, end with line containing only "."
C: .
```

```
C: QUIT
S: 554 no valid recipients
S: 221 goodbye
```

4. Security Considerations

    This RFC does not discuss security issues and is not believed to
    raise any security issues not endemic in electronic mail and present
    in fully conforming implementations of [1].

5. Acknowledgements

    This document is based on the SMTP service extension model presented
    in RFC 1425. Marshall Rose's description of SMTP command pipelining
    in his book "The Internet Message" also served as a source of
    inspiration for this extension.

6. References

    [1] Postel, J., "Simple Mail Transfer Protocol", STD 10
         RFC 821, USC/Information Sciences Institute, August
         1982.

    [2] Klensin, J., Freed, N., Rose, M., Stefferud, E.,
         and D. Crocker, "SMTP Service Extensions", RFC 1651,
         MCI, Innosoft, Dover Beach Consulting, Inc.,
         Network Management Associates, Inc., Silicon Graphics,
         Inc., July 1994.

7. Author's Address

    Ned Freed
    Innosoft International, Inc.
    1050 East Garvey Avenue South
    West Covina, CA 91790
    USA

    Phone: +1 818 919 3600
    Fax: +1 818 919 3614
    EMail: ned@innosoft.com

```