Compressed TCP/IP−Sessions using SSH−like tools

Table of Contents

Compressed TCP/IP−Sessions using SSH−like tools

Sebastian Schreiber <Schreib@SySS.de>

2.2.2000

---

1. Introduction

In the past, we used to compress files in order to save disk space. Today, disk space is cheap − but bandwidth is limited. By compressing data streams, you achieve two goals:

1) You save bandwidth/transfered volume (that is important if you have to pay for traffic or if your network is loaded.).

2) Speeding up low−bandwidth connections (Modem, GSM, ISDN).

This HowTo explains how to save both bandwith and connection time by using tools like SSH1, SSH2, OpenSSH or LSH.

---

2. Compressing HTTP/FTP,...

My office is connected with a 64KBit ISDN line to the internet, so the maximum transfer rate is about 7K/s. You can speed up the connection by compressing it: when I download files, Netscape shows up a transfer rate of up to 40K/s (Logfiles are compressable by factor 15). SSH is a tool that is mainly designed to build up secure connections over unsecured networks. Further more, SSH is able to compress connections and to do port forwarding (like rinetd or redir). So it is the appropriate tool to compress any simple TCP/IP connection. "Simple" means, that only one TCP−connection is opened. An FTP−connections or the connection between M$−Outlook and MS−Exchange are not simple as several connections are established. SSH uses the

LempleZiv (LZ77) compression algorithm – so you will achieve the same high compression rate as winzip/pkzip. In order to compress all HTTP−connections from my intranet to the internet, I just have to execute one command on my dial−in machine:

```
ssh −l <login ID> <hostname> −C −L8080:<proxy_at_ISP>:80 −f sleep 10000
```

`<hostname>` = host that is located at my ISP. SSH−access is required.

`<login ID>` = my login−ID on `<hostname>`

`<proxy_at_ISP>` =the web proxy of my ISP

My browser is configured to use `localhost:8080` as proxy. My laptop connects to the same socket. The connection is compressed and forwarded to the real proxy by SSH. The infrastructure looks like:

```
                  64KBit ISDN
My PC--------------------------------A PC (Unix/Linux/Win-NT) at my ISP
SSH-Client         compressed        SSH-Server, Port 22
Port 8080                              |
  |                                    |
  |                                    |
  |                                    |
  |10MBit Ethernet                     |100MBit
  |not compressed                      |not compressed
  |                                    |
  |                                    |
My second PC                         ISP's WWW-proxy
with Netscape,...                    Port 80
(Laptop)
```

---

3. Compressing Email

3.1 Incoming Emails (POP3, IMAP4)

Most people fetch their email from the mailserver via POP3. POP3 is a protocol with many disadvantages:

1. POP3 transfers password in clear text. (There are SSL−implementations of POP/IMAP and a challenge/response authentication, defined in RFC−2095/2195).
2. POP3 causes much protocol overhead: first the client requests a message than the server sends the message. After that the client requests the transferred article to be deleted. The server confirms the deletion. After that the server is ready for the next transaction. So 4 transactions are needed for each email.
3. POP3 transfers the mails without compression although email is highly compressible (factor=3.5).

You could compress POP3 by forwarding localhost:110 through a compressed connection to your ISP's POP3−socket. After that you have to tell your mail client to connect to localhost:110 in order to download mail. That secures and speeds up the connection −− but the download time still suffers from the POP3−inherent protocol overhead.

It makes sense to substitute POP3 by a more efficient protocol. The idea is to download the entire mailbox at once without generating protocol overhead. Furthermore it makes sense to compress the connections. The appropriate tool which offers both features is SCP. You can download your mail−file like this:

```
scp −C −l loginId:/var/spool/mail/loginid /tmp/newmail
```

But there is a problem: what happens if a new email arrives at the server during the download of your mailbox? The new mail would be lost. Therefore it makes more sense to use the following commands:

```
ssh −l loginid mailserver −f mv /var/spool/mail/loginid
/tmp/loginid_fetchme

scp −C −l loginid:/tmp/my_new_mail /tmp/loginid_fetchme
```

A move (mv) is a elementary operation, so you won't get into truble if you receive new mail during the execution of the comands. But if the mail server directories /tmp/ and /var/spool/mail are not on the same disc you might get problems. A solution is to create a lockfile on the server before you execute the mv: `touch /var/spool/mail/loginid.lock`. You should remove it, after that. A better solution is to move the file loginid in the same directory:

```
ssh −l loginid mailserver −f mv /var/spool/mail/loginid
/var/spool/mail/loginid_fetchme
```

After that you can use formail instead of procmail in order to filter /tmp/newmail into the right folder(s): `formail −s procmail < /tmp/newmail`

3.2 Outgoing Email (SMTP)

You send email over compresses and encrypted SSH−connections, in order to:

- Save network traffic
- Secure the connection (This does not make sense, if the mail is transported over untrusted networks, later.)
- Authenticate the sender. Many mail servers deny mail relaying in order to prevent abuse. If you send an email over an SSH−connection, the remote mail server (i.e. sendmail or MS−exchange) thinks to be connected, locally.

If you have SSH−access on the mail server, you need the following command:

```
ssh −C −l loginid mailserver −L2525:mailserver:25
```

If you don't have SSH−access on the mail server but to a server that is allowed to use your mail server as relay, the command is:

```
ssh −C −l loginid other_server −L2525:mailserver:25
```

After that you can configure your mail client (or mail server: see "smarthost") to send out mails to localhost port 2525.

4. Thoughts about performance.

Of course compression/encryption takes CPU time. It turned out that an old Pentium−133 is able to encrypt and compress about 1GB/hour −− that's quite a lot. If you compile SSH with the option "−−with−none" you can tell SSH to use no encryption. That saves a little performance. Here is a comprise between several download methods (during the test, a noncompressed 6MB−file was transfered from a 133MHz−Pentium−1 to a 233MHz Pentium2 laptop over a 10MBit ethernet without other load).

	FTP	encrypted	compressed	compressed & encrypted
Elapsed Time	|7.6s	26s	9s	23s
Throughput	790K/s	232K/s	320K/s	264K/s
Compression Factor	1	1	3.8	3.8

5. Greetings

Thanks to Harald König <koenig@tat.physik.uni−tuebingen.de>, who used rcp in order to download complete mailboxes. The latest version of this howto is available on `http://www.syss.de/howto.`