Long E-mails and Internet Connections

Merciadri Luca
Luca.Merciadri@student.ulg.ac.be

Abstract. Fetching e-mails is a daily task. However, when an e-mail size overcomes a given threshold, e-mail communication becomes more difficult, because slow. This is principally due to the outgoing speed of cheap Internet connections. We propose here a mechanism which limits this problem.

Keywords: E-mails.

1 Introduction

Fetching e-mails is a daily task. E-mail communication has become a *real alternative* to traditional communication medias, which are principally represented by (mobile) phones (through SMS or calls). However, when an e-mail size overcomes a given threshold, e-mail communication becomes more difficult, because slow. This is principally due to the outgoing speed of cheap Internet connections. We propose here a mechanism which limits this problem.

2 Context

A traditional way to communicate through e-mails is to choose to `ANSWER` to a read e-mail. This can be achieved on various e-mail clients (with or without a WWW GUI interface). Using this very useful method, one's e-mail can become quickly long, thus having a big size.

The phenomenon (which was explained in Section 1) only appears when, once an e-mail has been sent, this e-mail is reused many times to continue a given discussion related to the subject of the first mail. We only consider two persons sending e-mails using `POP` and `SMTP` protocols, that is, the most used protocols because of their simplicity and reputation.

In a more formal way, let's take two persons P_i ($i \in \{1,2\}$) communicating through e-mail. One of these P_i, say P_2, sends a first e-mail to the other (P_1). Once this e-mail has been read, P_1 answers for the first time (to P_2), and the reply is thus sent, then received, by P_2. Such a transaction is equivalent to *two sessions*. A *session* $s_j, j \in \mathcal{S}$, is thus successively defined by an e-mail reception, reading, replying and sending (by only one person, as shown before), the $\mathcal{S}$ set being the set of all the sessions.

If at least one of the P_i has to communicate regularly with the other P_i about the same precise subject, each of these P_i's e-mail clients will, when receiving an e-mail of the other P_i, attach the rest of the e-mail to the current e-mail. That is very useful in an e-mail conversation to know precisely what one is answering to, or even to proof other' words.

It is clear that if

1. $|\mathcal{S}| \gg$ (that is, if the cardinal of the set of all the sessions is bigger and bigger, the number of sessions being bigger and bigger), the fetching of the s-related e-mails will take more and more time,
2. $|\mathcal{S}| \pmod 2 = 0$, both P_i are replying to these e-mails. We here consider an infinite conversation (that is, $\mathcal{S}$ is an infinite set at a given point, but it keeps being finite for our examples).

We consider two extreme cases:

1. A session is formed by an everage amount of 10 words, such as

```
What do you think about this? I really like it.
```

2. A session is formed by an average amount of 1000 words, such as 100 times the given sentence.

We illustrate the phenomenon for both cases at Table 1, p.2 with a connection having a theoretical speed of 4096Kbps (2048 Kbps in practical measures) for Reception, and 256 Kbps (resp. 32.25 Kbps in pratical measures) for Sending. This connection is in Belgium, and no other task is performed during the test. Values of $|\mathcal{S}|$ are rounded to be in $\mathbb{Z}$.

2.1 Results

		(s)		# Sessions	
Size of the e-mail	Fetching time	Sending time	Total session time	$\vert\mathcal{S}\vert$ (10 wds. avg.)	$\vert\mathcal{S}\vert$ (1000 wds. avg.)
⋮	⋮	⋮	⋮	⋮	(N.A., too few words)
2 Ko (16 Kb)	0.0078125	0.4961240	0.5039365	2	⋮
10 Ko (80 Kb)	0.0390625	2.4615385	2.5006010	10	⋮
50 Ko (400 Kb)	0.1953125	12.3076923	12.5030048	50	⋮
100 Ko (800 Kb)	0.3906250	24.6153846	25.0060096	100	1
170 Ko (1360 Kb)	0.6640625	41.8461538	42.5102163	160	$1.7 \approx 1$
200 Ko (1600 Kb)	0.781250	49.2307692	50.0120192	200	2
500 Ko (4000 Kb)	1.9531250	123.076923	125.030048	500	5
⋮	⋮	⋮	⋮	⋮	⋮

Fig. 1. Results of e-mail sessions.

3 Interpretation

One can clearly notice that, according to Table 1,

1. If the average amount of words in a session is equal to 10, the sending time of the e-mail becomes too big (and is considered *very slow*) after 10 sessions,
2. If the average amount of words in a session is equal to 1000, the sending time of the e-mail becomes too big (and is considered *very slow*) after 0.1 session(!)

These remarks have sense, as e-mail is a quick communication way, and waiting more than 2 seconds to send an e-mail is too much. Furthermore, sending time of an e-mail handicaps total session time, which leads to a slower communication.

4 Proposition

Whatever the average number of words in a session, and according to the results of Table 1, a proposition has to be developed in order to lower

1. the sending time,
2. (and) the fetching time

of such e-mails. The fetching time is a little bit less important, as there needs to be at least 500 10-words-composed sessions (resp. 5 1000-words-composed sessions) to make fetching time as much annoying as sending time.

On one hand, when P_2 is replying to P_1, P_2 does not *need* to *send* the *whole* e-mail (that is, the current answer, and the quotation of both old answers) to its `SMTP` server.

On the other hand, when P_1 receives P_2's reply to P_1's e-mail, it would still be appreciable for P_1 to *receive* the *complete* e-mail.

Based on these two assumptions, we now give a method which would enhance the speed of delivery of such e-mails: output and input speeds.

4.1 POP Server As a Quotation Gateway

We here consider arbitrarily one of the P_i, $i \in \{1, 2\}$. Let's take P_2. If P_2

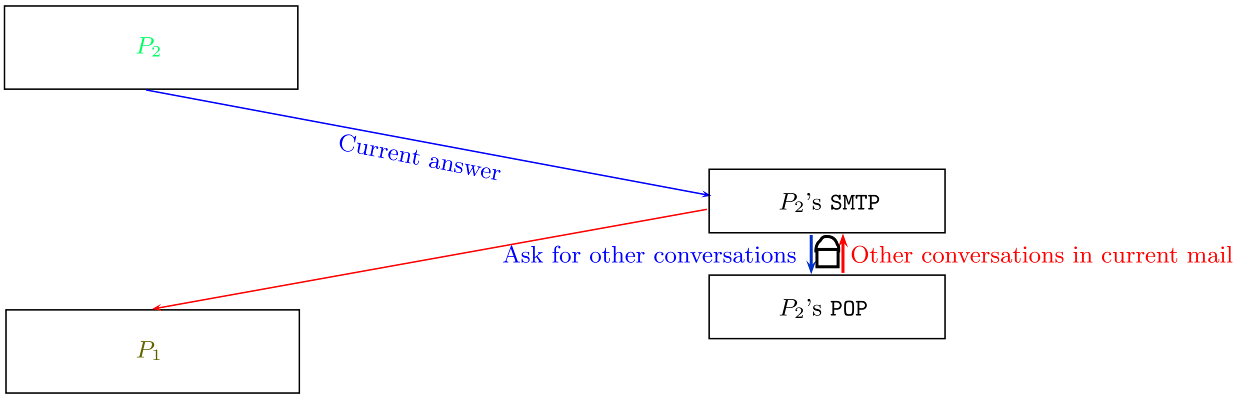
1. has a `POP` account, *and* that
2. sees his e-mail client not being configured to automatically delete fetched messages from the server,

an efficient way to communicate can be considered; it is schematized at Figure 4.1, p. 3. It would greatly enhance the replying time of P_2, still providing P_1 with the whole message.

If the second condition is not fulfilled, the method can be adapted easily. The first condition is always verified, as P_2 is assumed to be only using a `POP` server for his e-mails' fetching.

Completey (*i.e.* from the beginning to the end) using `SMTP` server as a quotation gateway would not be a good idea, as a `SMTP` server is made to *send* e-mails, not to hold them.

The connection from P_2 to P_2's `POP` server could be achieved easily and securely with an authentification, even if P_2's `POP` and P_2's `SMTP` servers are not on the same group of machines.


P_2
Current answer
P_2's SMTP
Ask for other conversations
Other conversations in current mail
P_2's POP
P_1

The steps would thus be the following ones:

1. (1 time) One of the P_i (say P_2) sends his first message to the other P_i (say P_1), through P_2's `SMTP` server,
2. (1 time) The other (P_1) receives P_2's e-mail thanks to P_1's `POP` server,
3. If the e-mail which P_1 has just received, has a size near the defined threshold, a signal is sent to P_1's `SMTP` server,
4. P_1 replies to P_2 through P_1's `SMTP` server:
   (a) If the signal has been sent, every text in the e-mail, except current text (text of this session), will be sent and attached by P_1's `SMTP` server to P_2's new e-mail (from P_1), the current text being sent by P_1;
   (b) If the signal has not been sent, the communication keep using traditional methods until the next signal,
5. The process restarts until $P_1 \leftrightarrow P_2$ communication is finished (hypothetically, never).

5 Benefits

The main concepts which had to be respected using a given method were the following:

1. The P_i's are able to see in an opaque way (*i.e.* without changing anything in their habits) quotations of old messages in the same e-mail,
2. P_i's I/O times must be kept as small as possible.

These two objectives are respected with our proposition. However, we have not quantified yet how much such a method would be interesting for the P_i's. The main advantage of our method is that, to lower P_i's e-mails' replies' I/O times, it uses the servers' Internet connections, but *not* in an *outrageous* way.

For example, assuming e-mail replies' composition (*i.e.* appending of previous quotations and old text to the current message) tasks are time-negligible (as they only use CPU time, this assumption can be done as this *task* is *easy*), the time of a session expression, multiplied by a d constant, equals

$$d \cdot t_{\text{session}} := t_f + t_s,$$

can divided by a d factor, thus giving

$$\boxed{t_{\text{session}} = \frac{t_f + t_s}{d}}, \tag{1}$$

t_f and t_s being respectively the amounts of time needed to fetch and to send the complete e-mail. We define $d := d_f + d_s$, d_f and d_s being respectively the factors of fetching and sending, with

$$d_f = \frac{s_f}{2400} \tag{2}$$

$$d_s = \frac{s_s}{256}, \tag{3}$$

s_f and s_s being respectively the *practical* (*i.e.* measured, not theoretical) speeds of reception and sending of the servers.

It leads to a global expression

$$t_{\text{session}} = \frac{t_f + t_s}{\frac{s_f}{2400} + \frac{s_s}{256}}. \tag{4}$$

In this expression, if we let $s_f \to +\infty$ and $s_s \to +\infty$ (that is, servers' fetching and outgoing speeds are positively infinite), thus asking simultaneously

$$\lim_{s_f \to +\infty} d_f = +\infty, \tag{5}$$

$$\lim_{s_s \to +\infty} d_s = +\infty, \tag{6}$$

we can consider

$$\lim_{\substack{d_f \to +\infty, \\ d_s \to +\infty}} t_{\text{session}} = \lim_{\substack{d_f \to +\infty, \\ d_s \to +\infty}} \left(\frac{t_f + t_s}{\underbrace{\frac{s_f}{2400}}_{=d_f} + \underbrace{\frac{s_s}{256}}_{=d_s}} \right) = 0, \tag{7}$$

as $t_f > 0$ and $t_s > 0$, thus showing that the P_i's fetching and sending times won't be as limited as before.

6 Theoretical Results

If we let, for example, $s_f = 24\,000\,\text{Kbps}$ and $s_s = 25\,600\,\text{Kbps}$, we have

$$t_{\text{session}} = \frac{t_f + t_s}{10 + 100} = \frac{t_f + t_s}{110}; \tag{8}$$

that is, $d = 110$. In the case where this method was not used, we can consider s_f and s_s being respectively equal to the P_i's s_f and s_s speeds, as the transaction is limited by the P_i. It thus leads to

$$t_{\text{session}} = \frac{t_f + t_s}{2}, \tag{9}$$

and, as

$$\frac{110}{2} = 55, \tag{10}$$

it shows us that a server executing our method, and having $s_f = 24\,000\,\text{Kbps}$ and $s_s = 25\,600\,\text{Kbps}$, thus speeds being 10 times equal to P_i's ones, gives a 55 global improvement.

7 Conclusion

In Section 1, we introduced the problem. Context and results were given in Section 2. We then interpreted results in Section 3, and made a proposal in Section 4. Benefits were exposed in Section 5; they were illustrated by a theoretical example in Section 6. This proves that this method is effective, and working. It should next be implemented.