Mercurial
Distributed Concurrent Versions System

Philippe Wambeke

LoliGrUB

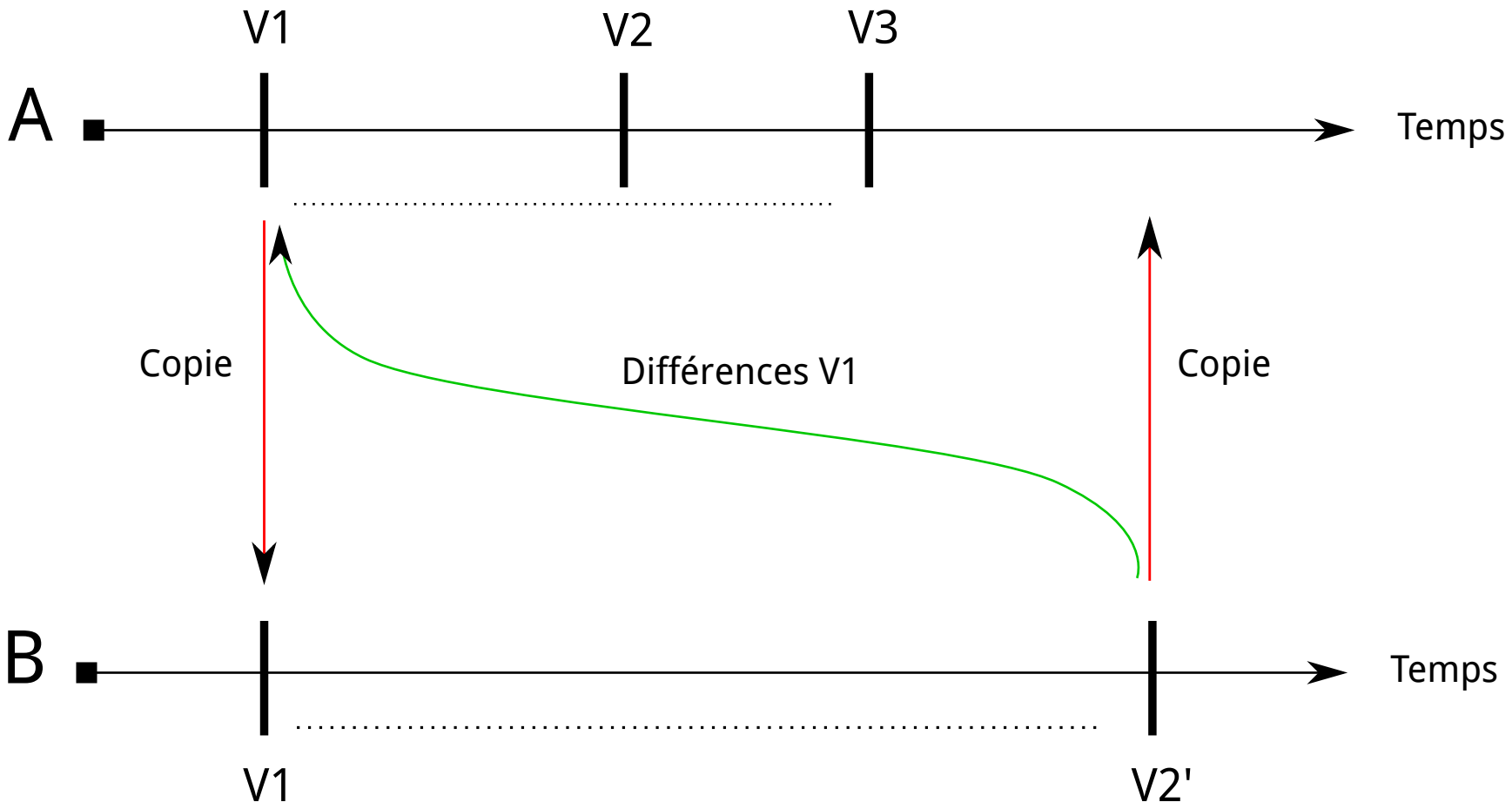
18 octobre 2014

Le problème à résoudre

La collaboration distante de plusieurs contributeurs à une tâche commune.

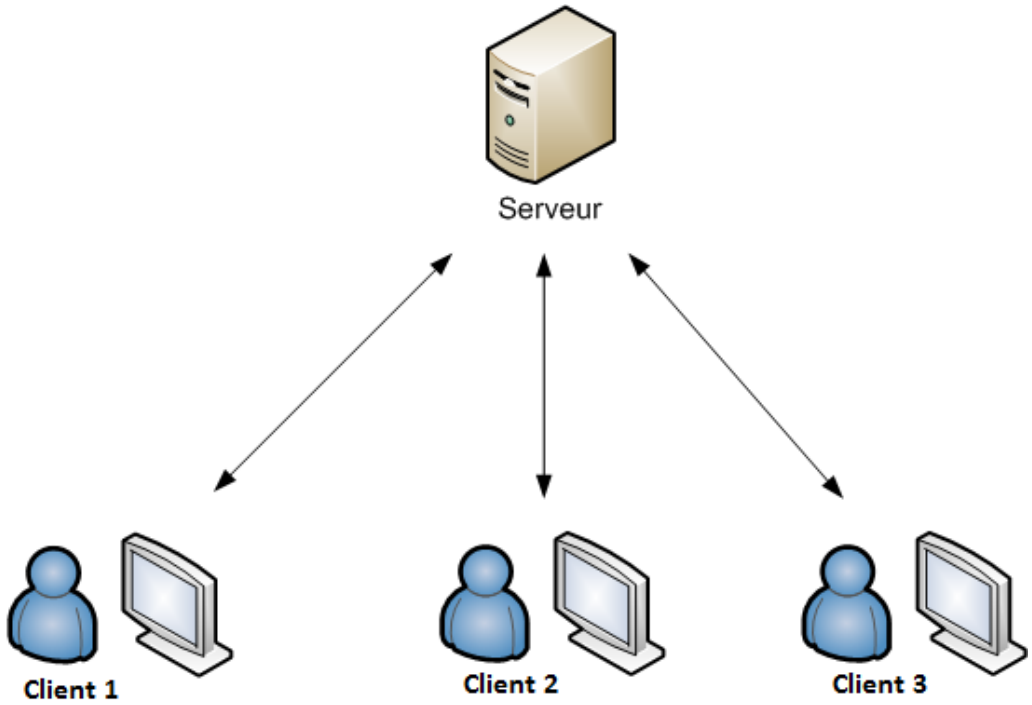
- Le développement d'un programme
- La rédaction d'un document
- L'écriture d'un roman
- La conception d'une recette de cuisine
- ...

Exemple simple



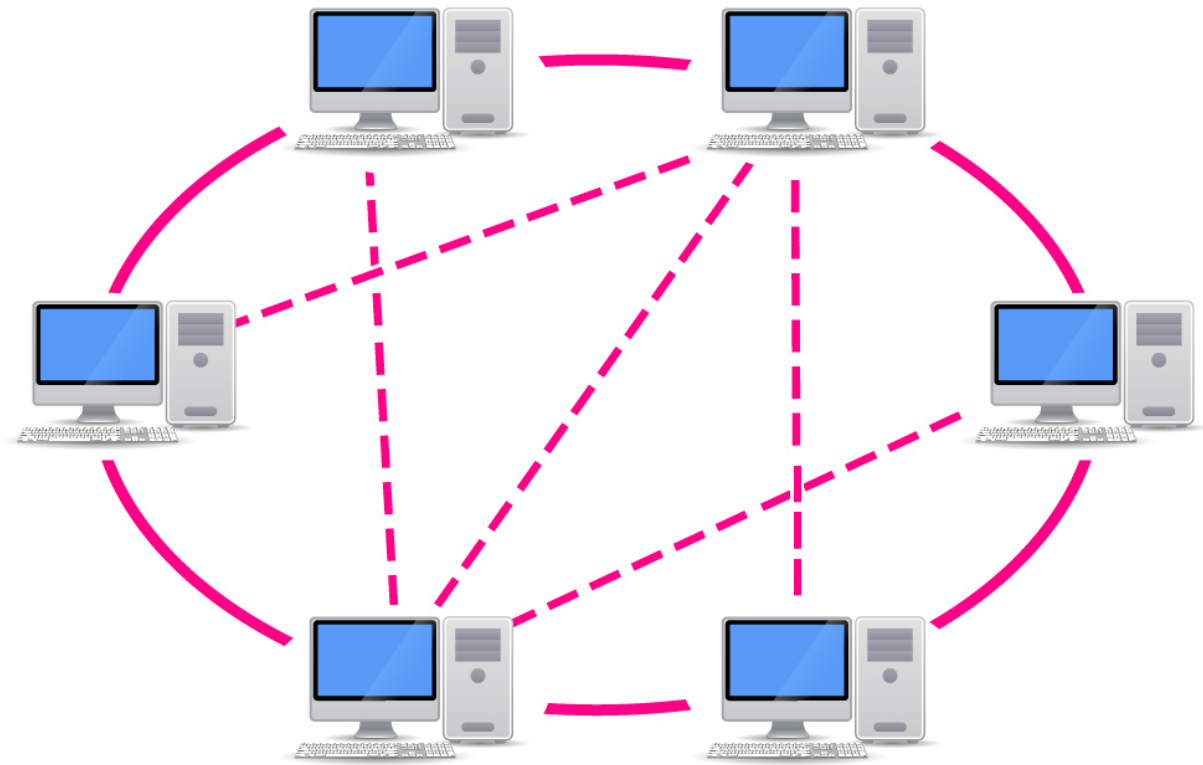
Les solutions

La solution centralisée :(



Les solutions

La solution décentralisée :D



Les solutions

Quelques exemples de solutions libres :

- Modèle client-serveur avec dépôt central :
  - CVS (Concurrent Version System)
  - SVN (Subversion)
- Modèle décentralisé :
  - Git
  - Mercurial !

Pourquoi Mercurial et pas Git ?

Pour plein de bonnes raisons :

Pourquoi Mercurial et pas Git ?

Pour plein de bonnes raisons :

- Je ne connais pas (bien) Git ;-)

Pourquoi Mercurial et pas Git ?

Pour plein de bonnes raisons :

- Je ne connais pas (bien) Git ;-)
- Mercurial est moins 'célèbre' que Git

Pourquoi Mercurial et pas Git ?

Pour plein de bonnes raisons :

- Je ne connais pas (bien) Git ;-)
- Mercurial est moins 'célèbre' que Git
- Mercurial est multi-plateformes depuis sa création (écrit en Python)

Pourquoi Mercurial et pas Git ?

Pour plein de bonnes raisons :

- Je ne connais pas (bien) Git ;-)
- Mercurial est moins 'célèbre' que Git
- Mercurial est multi-plateformes depuis sa création (écrit en Python)
- L'historique d'un dépôt Mercurial est gravé dans le marbre

Pourquoi Mercurial et pas Git ?

Pour plein de bonnes raisons :

- Je ne connais pas (bien) Git ;-)
- Mercurial est moins 'célèbre' que Git
- Mercurial est multi-plateformes depuis sa création (écrit en Python)
- L'historique d'un dépôt Mercurial est gravé dans le marbre
- Mercurial est simple à apprendre

D'où vient Mercurial ?

Le développement de Mercurial a commencé en même temps que celui de Git (en 2005), après le fiasco BitKeeper.

Linus Torvalds a développé Git.

Matt Mackall a développé Mercurial.

L'objectif était de disposer d'un outil de gestion de version simple, décentralisé, léger et puissant.

D'où vient Mercurial ?

Le développement de Mercurial a commencé en même temps que celui de Git (en 2005), après le fiasco BitKeeper.

Linus Torvalds a développé Git.

Matt Mackall a développé Mercurial.

L'objectif était de disposer d'un outil de gestion de version simple, décentralisé, léger et puissant.

Objectif atteint !

Qui utilise Mercurial ?

Quelques petits projets pas très connus, comme :

Qui utilise Mercurial ?

Quelques petits projets pas très connus, comme :

- Vim (l'éditeur ultime)

Qui utilise Mercurial ?

Quelques petits projets pas très connus, comme :

- Vim (l'éditeur ultime)
- ALSA (Gestion du son sous Linux)

Qui utilise Mercurial ?

Quelques petits projets pas très connus, comme :

- Vim (l'éditeur ultime)
- ALSA (Gestion du son sous Linux)
- OpenJDK (implémentation libre de Java)

Qui utilise Mercurial ?

Quelques petits projets pas très connus, comme :

- Vim (l'éditeur ultime)
- ALSA (Gestion du son sous Linux)
- OpenJDK (implémentation libre de Java)
- Dovecot (Serveur de mails)

Qui utilise Mercurial ?

Quelques petits projets pas très connus, comme :

- Vim (l'éditeur ultime)
- ALSA (Gestion du son sous Linux)
- OpenJDK (implémentation libre de Java)
- Dovecot (Serveur de mails)
- Python

Qui utilise Mercurial ?

Quelques petits projets pas très connus, comme :

- Vim (l'éditeur ultime)
- ALSA (Gestion du son sous Linux)
- OpenJDK (implémentation libre de Java)
- Dovecot (Serveur de mails)
- Python
- Firefox (tous les logiciels de la fondations Mozilla)
- ...

Installation

3 outils sont à installer pour utiliser confortablement Mercurial :

1. Mercurial
2. hgview (visualisation de l'historique en mode graphique)
3. kdiff3 (comparateur visuel)

Debian :

```
sudo apt-get install mercurial hgview kdiff3
```

Arch :

```
sudo pacman -S mercurial hgview kdiff3
```

Configuration minimale

Toute la configuration se fait dans un seul fichier : .hgrc

```
[ui]
username = wap <wap@wambeke.org>

[extensions]
hgext.graphlog =
hgext.extdiff =

[extdiff]
cmd.kdiff3 =

[merge-tools]
kdiff3.args = $base $local $other -o $output
```

Pour bien commencer

Première chose à faire : initialiser le dépôt :

```
hg init
```

Ajouter ensuite les fichiers à 'surveiller' :

```
hg add
```

Enregistrer les changements :

```
hg commit -m "Mon message de commit"
```

Connaître l'état de la copie de travail

Vérifier l'état de la copie de travail :

```
hg status
```

Consulter l'historique :

```
hg log
```

Diffuser le dépôt (1)

Faire un clône du dépôt (en local ou à distance) :

```
hg clone <dépôt_origine> <dépôt_cible>
```

Envoyer ce qu'il y a de nouveau :

```
hg push <dépôt_cible>
```

Recevoir ce qu'il y a de nouveau :

```
hg pull <dépôt_cible>
```

Diffuser le dépôt (2)

Préparer un "paquet" avec plusieurs révisions :

```
hg bundle --base rev FILE
```

Mettre à jour la copie de travail :

```
hg update [-r REV]
```

Gérer les branches

Lister les différentes "têtes" :

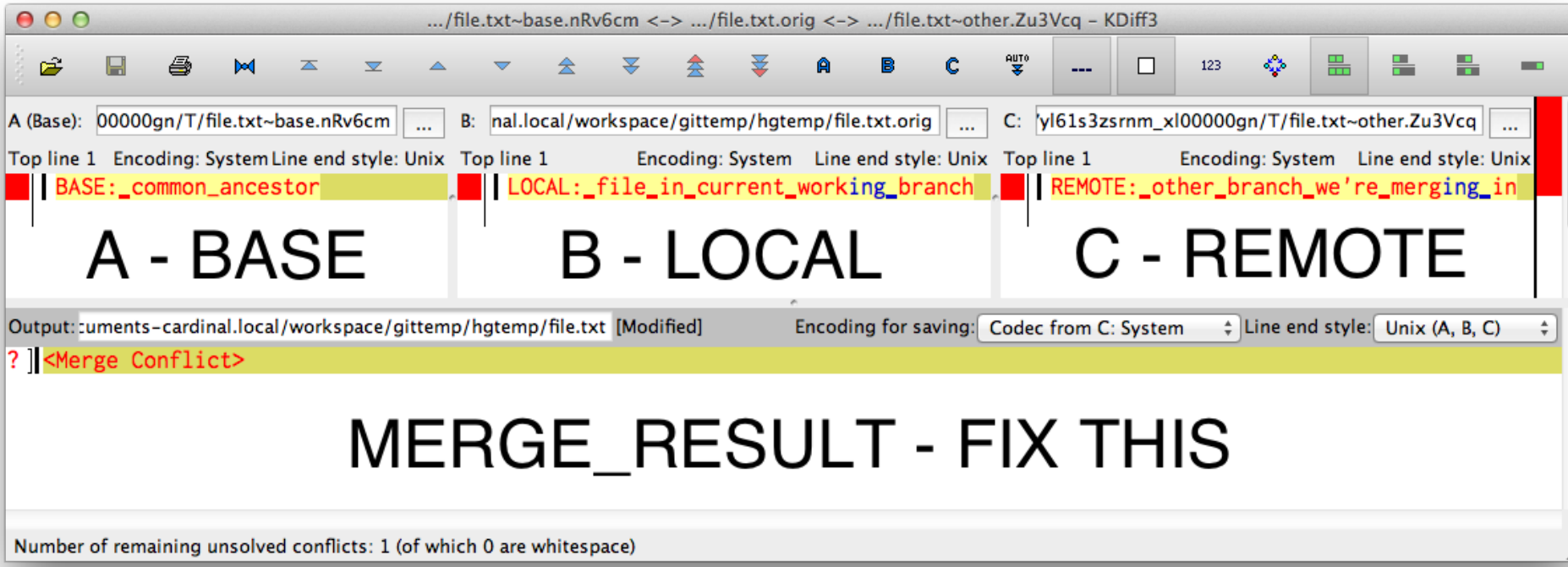
```
hg heads
```

Fusionner les branches :

```
hg merge [-r REV]
```

Gérer les conflits

Avec une bonne configuration du fichier .hgrc, la résolution des conflits est un jeu d'enfant grâce à KDiff3.



Quelques commandes pour la route

- hg remove : supprime un fichier du dépôt
- hg branch : nomme une branche
- hg branches : affiche les différentes branches du dépôt
- hg tag : pose une étiquette sur une révision
- hg tags : affiche les étiquettes du dépôt
- hg revert : revient à la version d'origine d'un fichier

Bonnes pratiques

- Rédigez des messages de commit (et de merge) significatifs, complets et précis

Bonnes pratiques

- Rédigez des messages de commit (et de merge) significatifs, complets et précis
- Faites des commit "atomiques"

Bonnes pratiques

- Rédigez des messages de commit (et de merge) significatifs, complets et précis
- Faites des commit "atomiques"
- Mercurial n'est pas un outil de backup !

Conclusion

Essayer Mercurial, c'est l'adopter !

Merci...

Questions ?