Prenez le contrôle de votre système d'impression

Jérôme Alet
alet@librelogiciel.com

Journée Méditerranéenne des Logiciels Libres
15 Novembre 2008
Sophia Antipolis

Quelques briques logicielles pour tirer vraiment partie de CUPS

- pkipplib
- pkpgcounter
- Tea4CUPS
- PyKota

Quelques rappels sur CUPS

- Logiciel serveur d'impression sous GPL
- Depuis 1997 (Easy Software Products)
- Plus évolué que LPD/LPR
- Plus facile à configurer que LPRng (interface web http://localhost:631)
- Serveur Internet Printing Protocol
- Autodétection serveurs et imprimantes
- Répartition de charge
- Pas besoin de pilote sur les clients
- Fonctionne sur tout système *nix
  - ➔ Standard de fait sur :
    - GNU/Linux
    - Mac OS X

pkipplib

Présentation de la librairie pkipplib
Support du protocole IPP pour Python

Description de pkipplib

- Librairie pour le langage Python
- 'pk' => extraite à l'origine de PyKota
- Internet Printing Protocol :
  - Novell + Xerox + IETF (ex RFC2910, 2911)
  - Protocole d'impression « puissant » :
    - Utilise HTTP 1.1 pour le transport :
      - Pas limité aux réseaux locaux
      - Contrôle d'accès
      - Authentification
      - Cryptage
      - ...
    - Permet de soumettre et annuler des travaux
    - Permet aussi d'interroger les imprimantes

Fonctionnalités de pkipplib

- Encodage et décodage de requêtes IPP
  - IPP est un protocole « binaire » :
    1. Version d'IPP (en général 1.1)
    2. Code opération (ex: 2 pour imprimer)
    3. Identifiant de requête, ou 1
    4. Attributs obligatoires (type + valeur)
    5. Attributs optionnels (type + valeur)
    6. Marqueur de fin des attributs
    7. Données, par exemple document PostScript
- Interaction avec des serveurs IPP :
  - Imprimantes
  - Novell iPrint
  - CUPS (tous les *nix)
  - Autres...

Exemple : imprimer un fichier PDF

```
#! /usr/bin/env python
# -*- coding: utf-8 -*-

# Importe la librairie :
from pkipplib import pkipplib

# Instancie une connexion à un serveur CUPS
cups = pkipplib.CUPS() # Par défaut http://localhost:631
# Crée la requête IPP adéquate :
req = cups.newRequest(operationid=pkipplib.IPP_PRINT_JOB)
# Positionne les paramètres nécessaires :
req.operation["requesting-user-name"] = ("nameWithoutLanguage", "jerome")
req.operation["printer-uri"] = ("uri", cups.identifierToURI("printers", "HP2100"))
req.operation["document-format"] = ("mimeMediaType", "application/pdf")
# Place un fichier PDF dans la requête
inputfile = open("demo.pdf", "rb")
req.data = inputfile.read()
inputfile.close()
# Et envoie la requête d'impression à CUPS.
response = cups.doRequest(req)
```

Avantages / Inconvénients de pkipplib

- Avantages :
  - 100% Python :
    - Pas besoin de la librairie cliente CUPS
    - → Fonctionne sur tous les OS supportant Python
  - Simple à utiliser
  - Permet de gérer les abonnements IPP

- Inconvénients :
  - Supporte seulement les sockets IP, pas celles du domaine unix
  - API non finalisée. Version 0.07, contributions bienvenues :-)

pkpgcounter

Présentation de pkpgcounter
Analyseur de documents

Description de pkpgcounter

- Outil en mode ligne de commande
- Librairie pour le language Python
- 'pk' => extrait à l'origine de PyKota

- Deux modes de fonctionnement :
  - Simple comptage de pages
  - Calcul du taux de couverture d'encre

Mode comptage de pages

1. Autodétection du Langage de Description de Page (PDL)
2. Activation du module d'analyse adéquat
3. Analyse directe du contenu :
   - ±20 Parsers internes
   - Cas « particulier » PostScript
   - Cas particulier « .doc »
4. Renvoi du nombre de pages calculé

Mode calcul du taux de couverture d'encre

1. Autodétection du PDL
2. Activation du module d'analyse adéquat
3. Conversion vers TIFF (outils externes)
4. Analyse selon l'espace colorimétrique :
   - BW
   - RGB
   - CMY
   - CMYK
   - GC : espace colorimétrique « maison »
5. Pour chaque page, résultat de type :

C: 0.873% M: 0.775% Y: 0.576% K: 2.883%

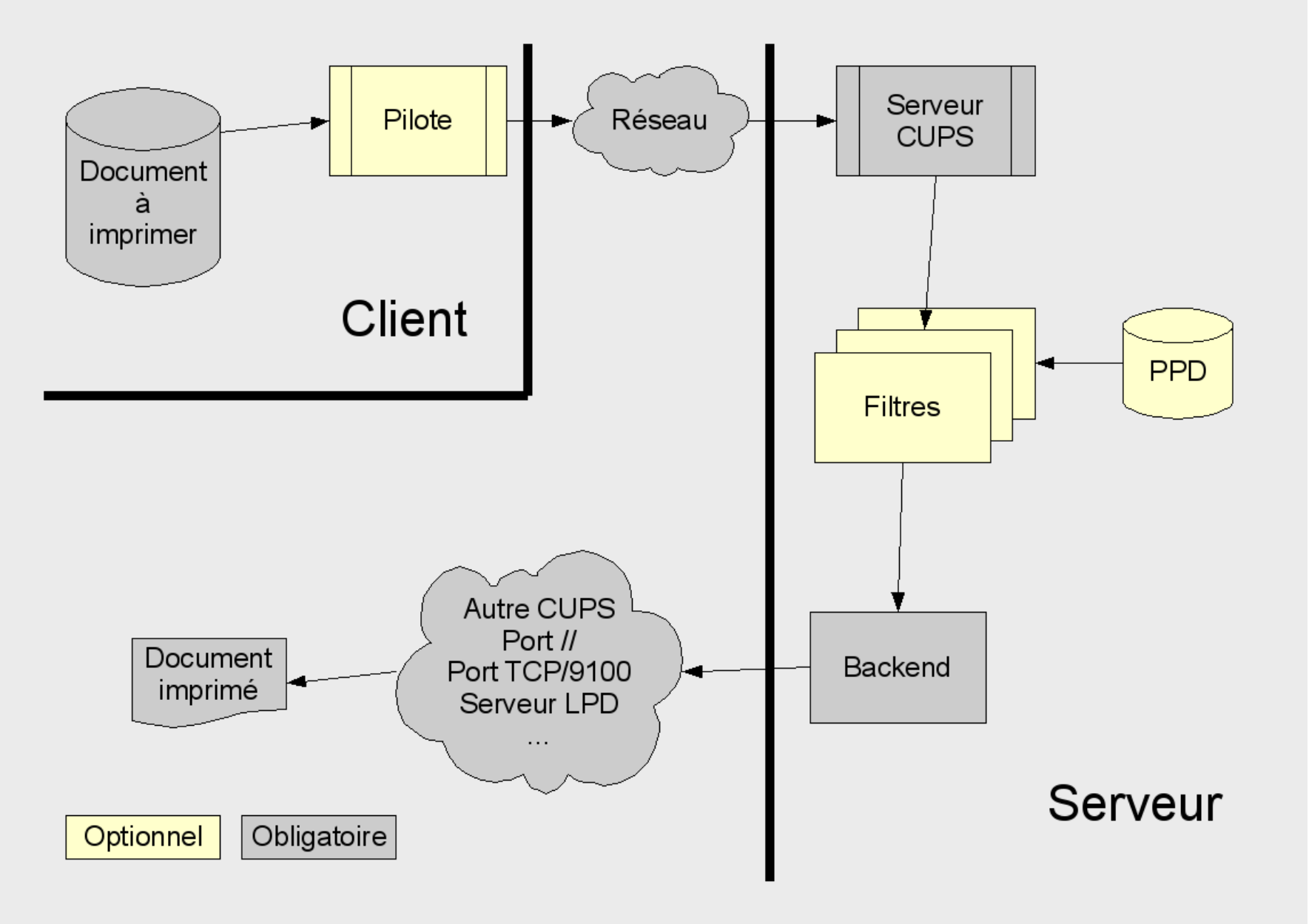
Fonctionnalités manquantes

- Formats de fichiers :
  - Les plus courants sont déjà supportés
  - Formats entièrement propriétaires
  - Formats partiellement propriétaires
- Détection du mode d'impression :
  - Duplex/Simplex
  - Taille des pages
  - N-up
  - Longueur du papier (traceurs)
- Taux de couverture d'encre : manque ***vraiment*** pour les pilotes Epson

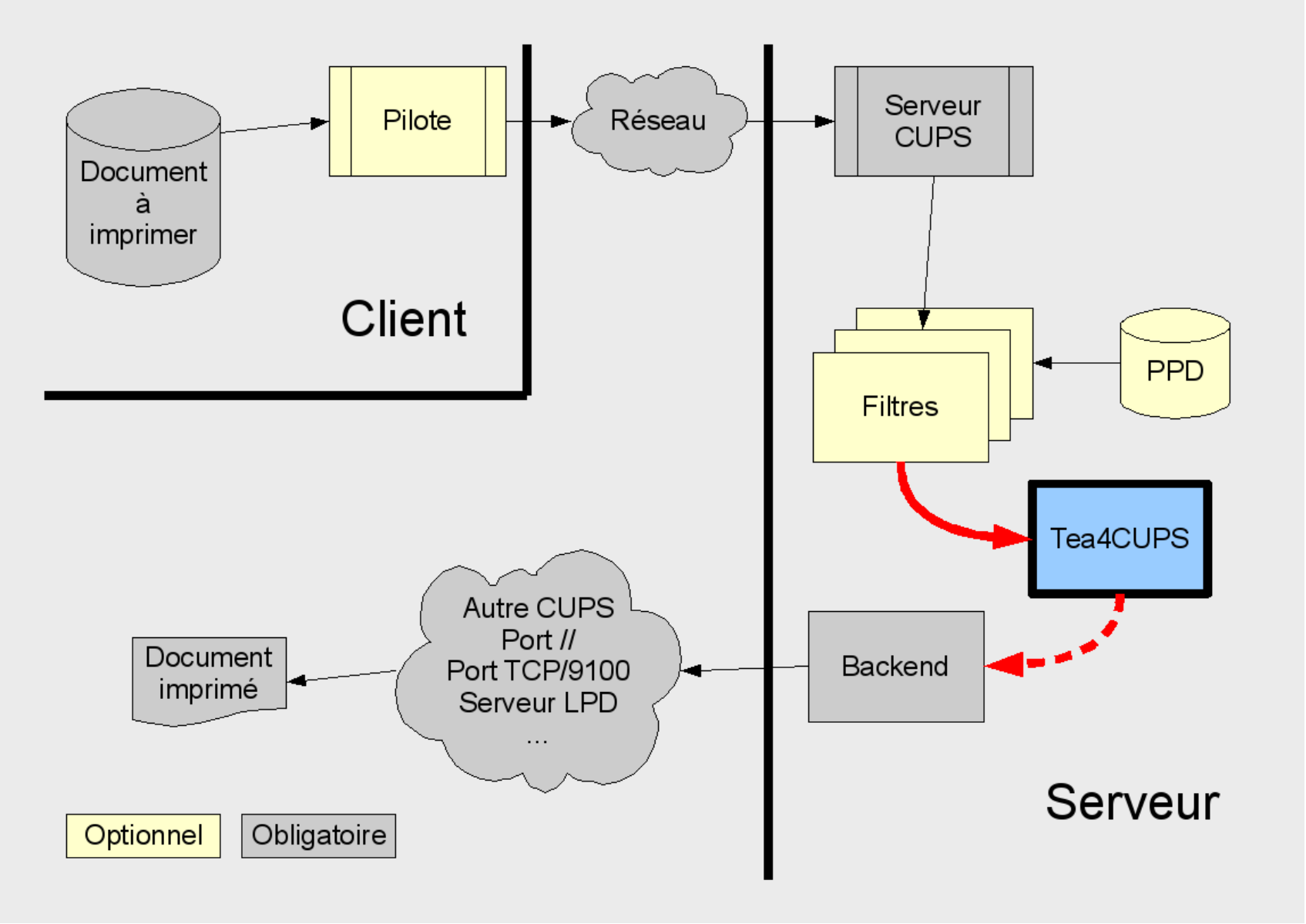
Tea4CUPS

Présentation de Tea4CUPS
« Couteau Suisse » de l'admin CUPS

Flux d'impression CUPS



Flux d'impression avec Tea4CUPS



Description de Tea4CUPS

- Backend CUPS (/usr/lib/cups/backend/)
- Inspiré de la commande '*tee*'
- Capture les flux d'impression
- Peut en filtrer le contenu
- Positionne des variables d'environnement
- Exécute des « prehooks »
- Exécute le backend CUPS réel
- Exécute des « posthooks »

tea4cups.conf

```
[global]
directory : /var/spool/tea4cups
debug : no
keepfiles : no
filter : /bin/grep -v "%%CreationDate:"

posthook_etat : echo Job $TEAJOBID imprimé (état $TEASTATUS)
                                        | smbclient -M $TEAUSERNAME
prehook_compta : echo $TEAPRINTERNAME $TEAJOBID $TEAUSERNAME
                          $TEABILLING `pkpgcounter $TEADATAFILE`
posthook_compta : /bin/cat >>/var/log/printaccounting.log

[MonImprimantePostScript]
prehook_rawpdf : /bin/cat $TEADATAFILE
       | /bin/su $TEAUSERNAME -c
       "ps2pdf - `/usr/bin/getent passwd $TEAUSERNAME
                  | /usr/bin/cut -f 6,6 -d :`/JOB-$TEAJOBID.pdf"
posthook_rawpdf : /bin/cat >>/tmp/logs_et_erreurs_ps2pdf.log
```

Possibilités (peu de limites)

- Archivage
- Publication
- Limitation & Multiplication
- Rétention & Libération (Hold & Release)
- Routage vers l'imprimante la plus proche
- Modification :
  - Ajout de fonds de pages, de logos et signatures, etc...
- Et pourquoi pas :
  - Domotique :

    lp -dcafetière expresso.recette
  - Trucs bizarres :

    lp -dscanner scanjob.description
    - acquisition, transformation, impression...
  - Tout ce qui vous passe par la tête

PyKota

Présentation de PyKota
Logiciel de quotas d'impression

Pourquoi ???

- Comptabilisation (monde de l'entreprise) :
  - Audit :

    acheté – stock – consommé = gaspillé
  - Planification :
    - Approvisionnements
    - Maintenance préventive
  - Facturation :
    - Par service, par utilisateur, par client, etc...

- Limitation (monde de l'éducation) :
  - Des coûts :
    - Consommables : papier, encre
    - Matériels : achat, maintenance
  - De l'impact environnemental :
    - Forêts (+ ou -), Eau++, produits chimiques++

    http://feuille-erable.org/accueil_cycle.htm

Description de PyKota

- ***Grossièrement*** PyKota combine :
  - pkipplib
  - pkpgcounter
  - Tea4CUPS
  - Un support de sauvegarde des données :
    - PostgreSQL
    - MySQL
    - LDAP (OpenLDAP, SunOne)
    - SQLite
  - Des interfaces utilisateur :
    - Ligne de commande : administration et interrogation
    - Web : interrogation seule
    - Natives graphiques OSD et/ou boîtes de dialogues : interaction avec utilisateur final
  - Et...

Description de PyKota (suite)

- VOTRE IMAGINATION :
  - Tous les points stratégiques sont SCRIPTABLES !
  - Pas de lien en dur entre compte d'impression et utilisateur système : on peut décompter par adresse IP ou MAC du client si on le désire, etc...
  - Interactions complexes possibles avec l'utilisateur final (avec add-on, ou pas) : information, confirmation, authentification, etc...

5 Modes de limitation des impressions

- Nombre de pages par imprimante :
  - 100 pages sur HP LaserJet 2100
  - 30 pages sur EPSON Stylus Color
- Nombre de crédits à consommer :
  - 1 page = x crédits sur HP 2100
  - 1 page = y crédits sur Stylus Color
  - + Coût par job (optionel)
  - + Coûts induits (hiérarchies d'imprimantes)
  - * Facteur de surtaxe du compte (+ ou -)
- Pas de limite, mais décompte effectué
- Pas de limite, aucun décompte effectué
- Impression interdite

2 Modes de comptabilisation

- Décompte matériel (interrogation directe des imprimantes) :
  - SNMP
  - HP PJL
  - Scripts externes

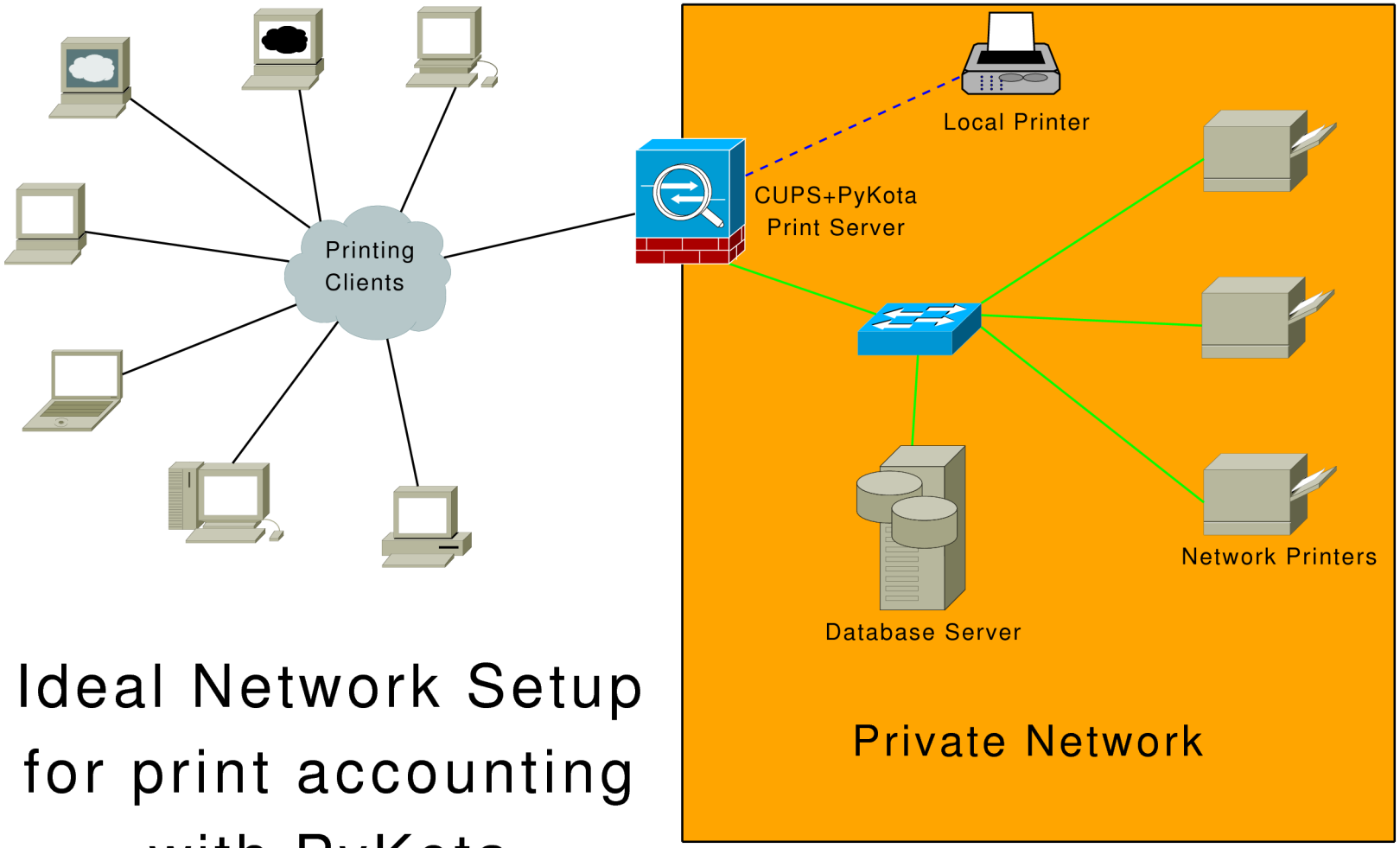
- Décompte logiciel :
  - pkpgcounter :
    - Calcul du nombre de pages
    - Calcul du taux de couverture d'encre
  - Scripts externes :
    - Calcul du nombre de pages

Fonctionnalités pratiques

- Création des imprimantes et comptes d'impression « à la volée »
- Imprimantes en mode transparent : examens...
- Taille maximale d'un job par imprimante
- Création de factures d'impression ou de certificats de remboursement au format PDF
- Exports CSV, XML...
- Groupes d'utilisateurs
- Groupes d'imprimantes imbricables :

  Laser -> HP,DELL -> LJ2100, LJ2200, 5110CN

Installation recommandée



Ideal Network Setup
for print accounting
with PyKota

Distribution et Financement

- Logiciels Libres
- Licence GNU GPL v3 ou + *(sauf Tea4CUPS en v2 ou +)*
- Téléchargement par subversion ou *.tar.gz*
- Sauf pour PyKota et Tea4CUPS :
  - Téléchargement par subversion gratuit
  - Téléchargement *.tar.gz*, *.deb*, *.rpm* « OFFICIELS » avec droit d'entrée de ***25 EUROS***, durée illimitée.
  - Mais bien sûr : la redistribution et la modification* sont autorisées.

* Je demande à ce qu'une version « officielle » modifiée ne contienne plus le mot « officiel » dans le numéro de version. **Ce n'est pas une obligation.**

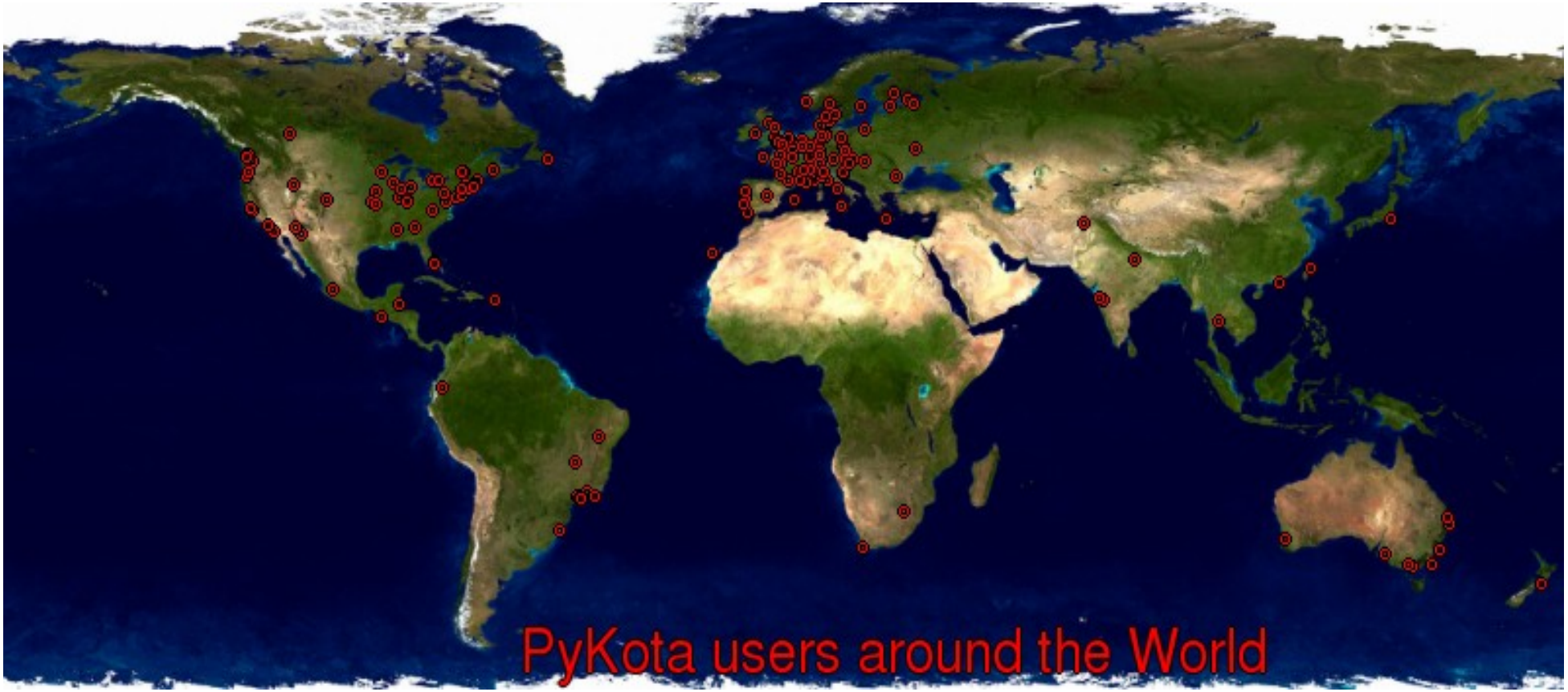
- Contrats d'assistance technique
- Installations guidées ou pilotées

Impact du mode de distribution

- Communauté peu « contributive »
- Possible d'encourager les contributions :
  - Cadeaux divers : livres, T-shirts...
  - Argent
  - ???
  - → MAIS : comment rester équitable ?

- Réactions :
  - Peu de critiques (1%)
  - Pas de redistribution tierce partie de paquets « Officiels », pourtant autorisée
  - Développements d'add-ons

Qui utilise PyKota ?



Des liens !

- Site Web :
  - *http://www.pykota.com*
- Bug Tracker :
  - *http://trac.pykota.com*
- IRC :
  - *#pykota* sur *irc.freenode.net*
- Listes de diffusion *@lists.pykota.com* :
  - *pykota-devel* (25+) : surtout « commits »
  - *pykota* (300+) : assistance technique
  - *pykota-support* (580+), lecture seule, utilisateurs enregistrés : annonces, correctifs, sécurité...