REbus

Un bus de communication facilitant la coopération entre outils d'analyse de sécurité

Philippe Biondi, Xavier Mehrenberger, Sarah Zennou

3 juin 2015 / SSTIC

AIRBUS GROUP

Plan

1. Pourquoi et comment REbus ?
2. Concepts de REbus
   - Vue d'ensemble
   - Exemple d'agent
   - Implémentations du bus
3. Ensembles d'agents REbus
4. Conclusion

Plan

Introduction - multitude de programmes spécialisés

Travaux d'analyse de sécurité

- Nombreux outils d'analyse spécialisés
- Chacun d'entre eux aide à résoudre une partie du problème
- Tâches souvent répétitives

REbus

- Outil développé pour résoudre le problème
- Licence BSD
- `https://bitbucket.org/iwseclabs/rebus`

Exemples de domaines d'analyse

- Reconnaissance sur un réseau [a]
- Analyse forensique de disques durs [b]
- Analyse de certificats
- Analyse de javascript & pages web
- Collecte d'information sur les menaces (*Threat Intelligence*)

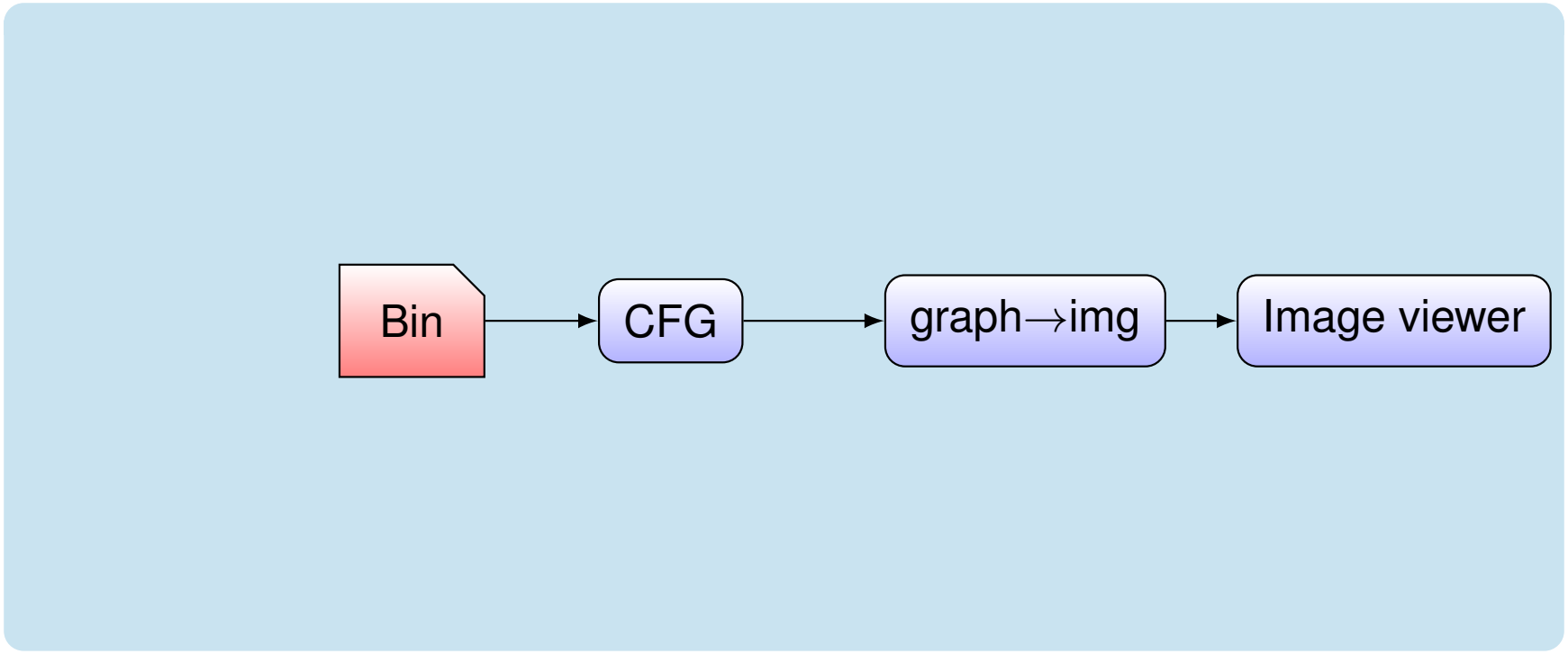
---

a. `https://bitbucket.org/iwseclabs/discobus`
b. `https://github.com/jahrome/DFIRbus`

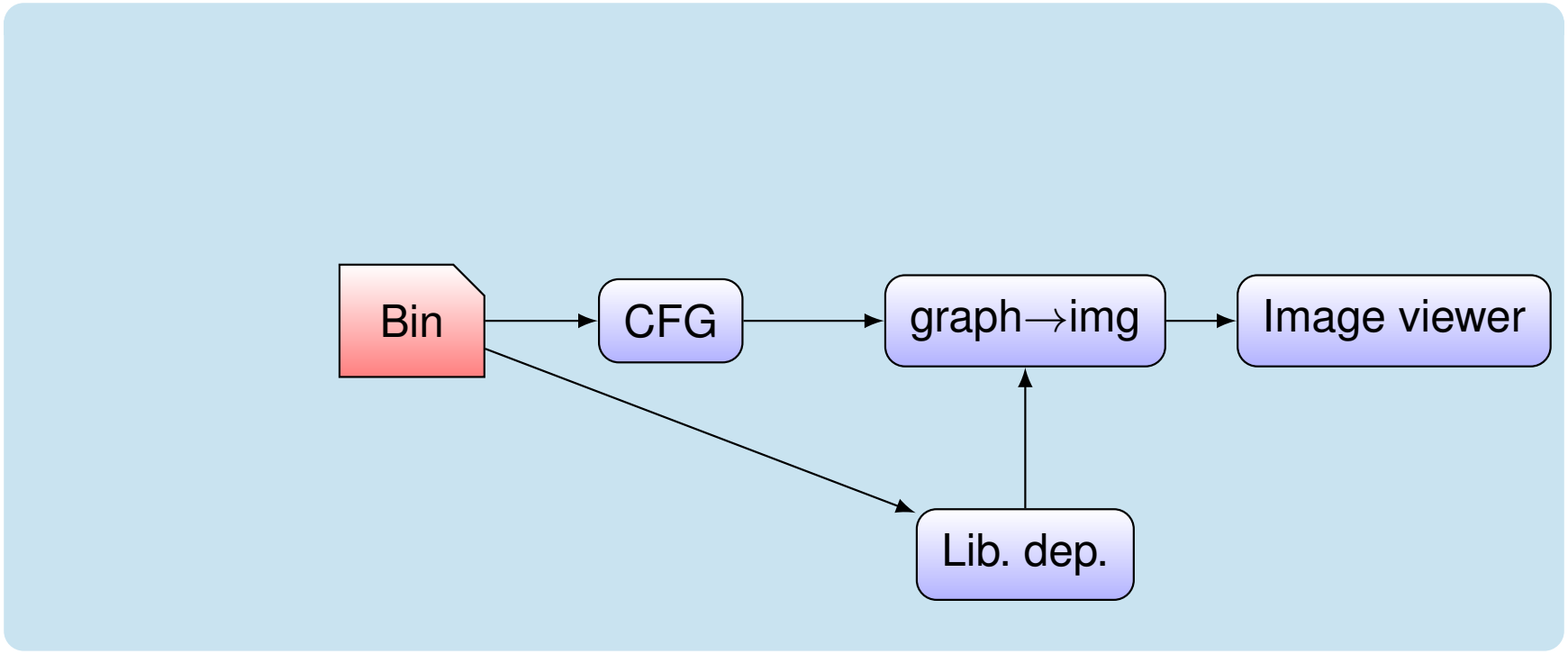
Écosystème d'outils d'analyse de programmes binaires

- Manipulation d'exécutables : Miasm, ElfEsteem, Amoco, metasm, ...
- Graphes : graphviz, Grandalf, ...
- Analyseurs statiques : libmagic, PEID, BAP, Bindiff, binwalk, outils maison, ...
- Outils interactifs : IDA, radare2, ...
- Sandboxes : Cuckoo, FireEye, ...
- Tests antivirus : VirusTotal, IRMA, ...
- Outils de classification
- Unpackers

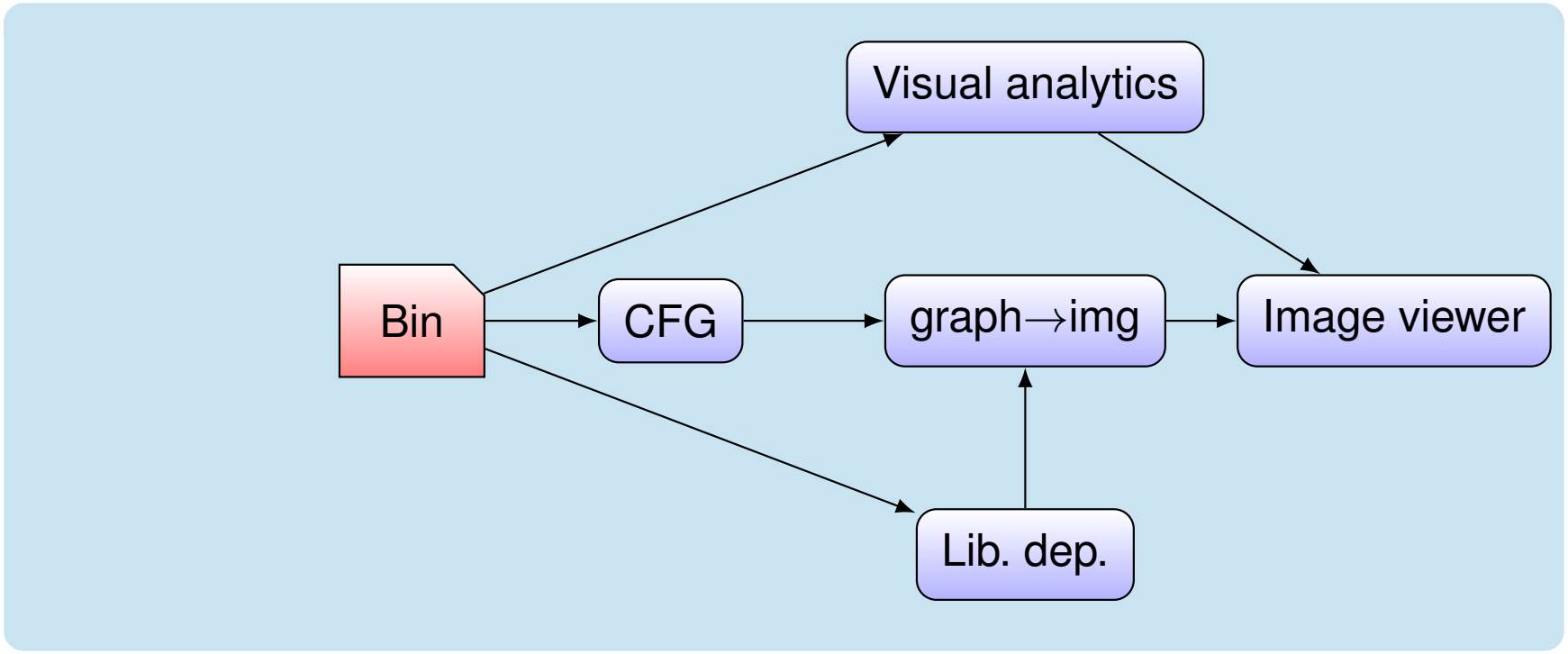
Exemple de workflow d'analyse



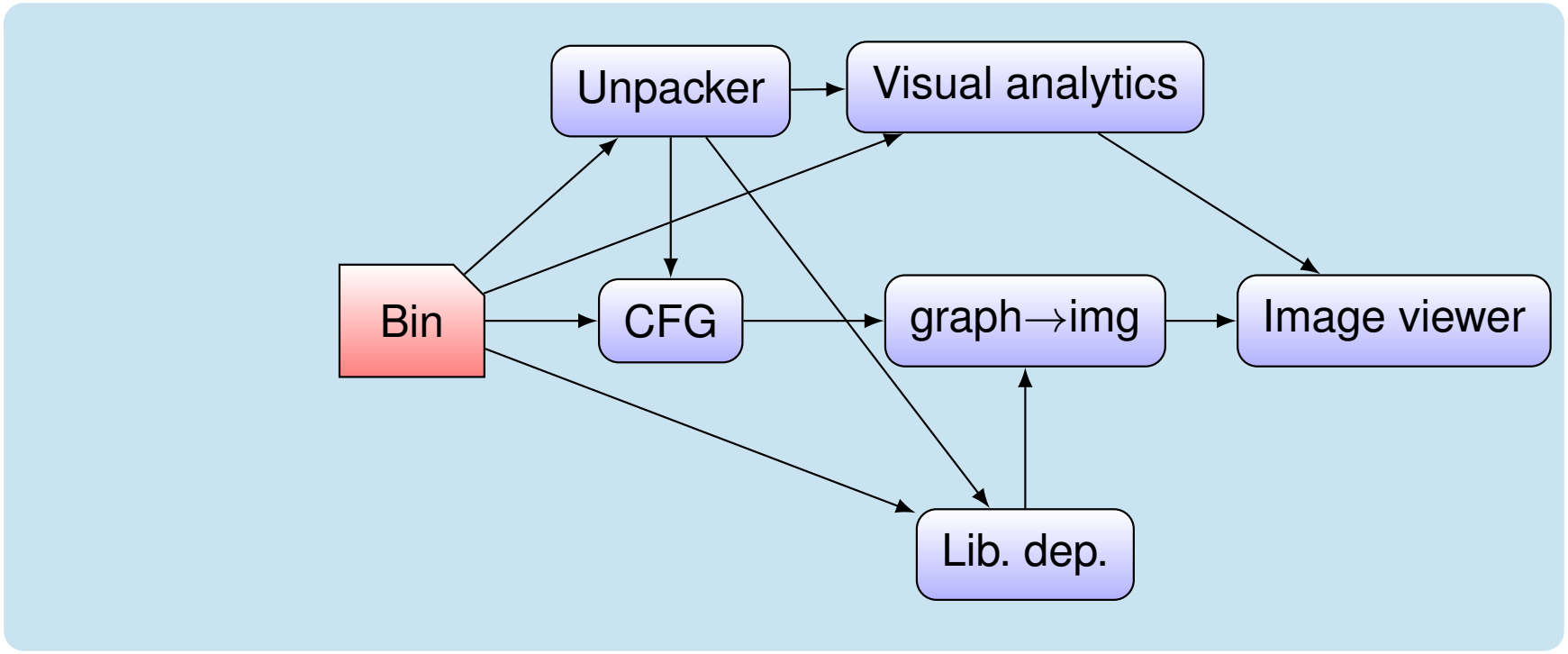
Exemple de workflow d'analyse



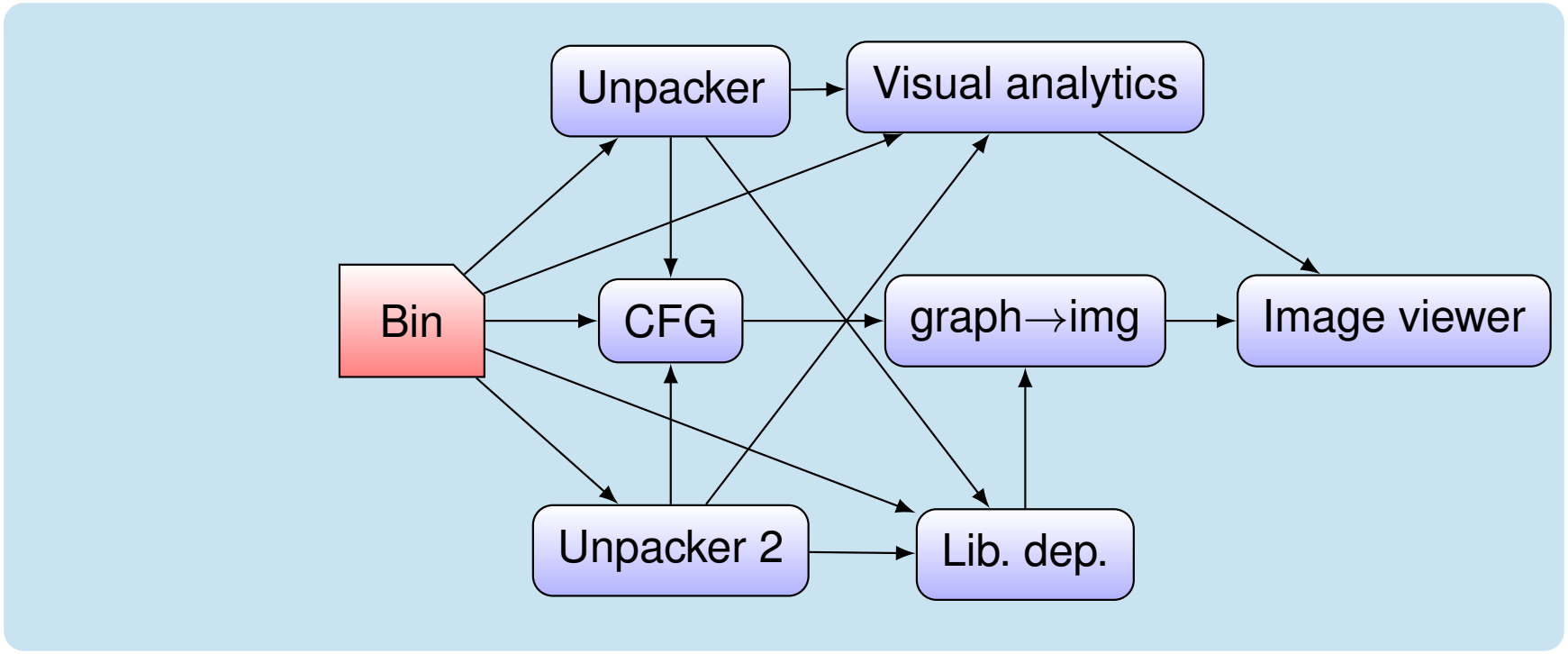
Exemple de workflow d'analyse



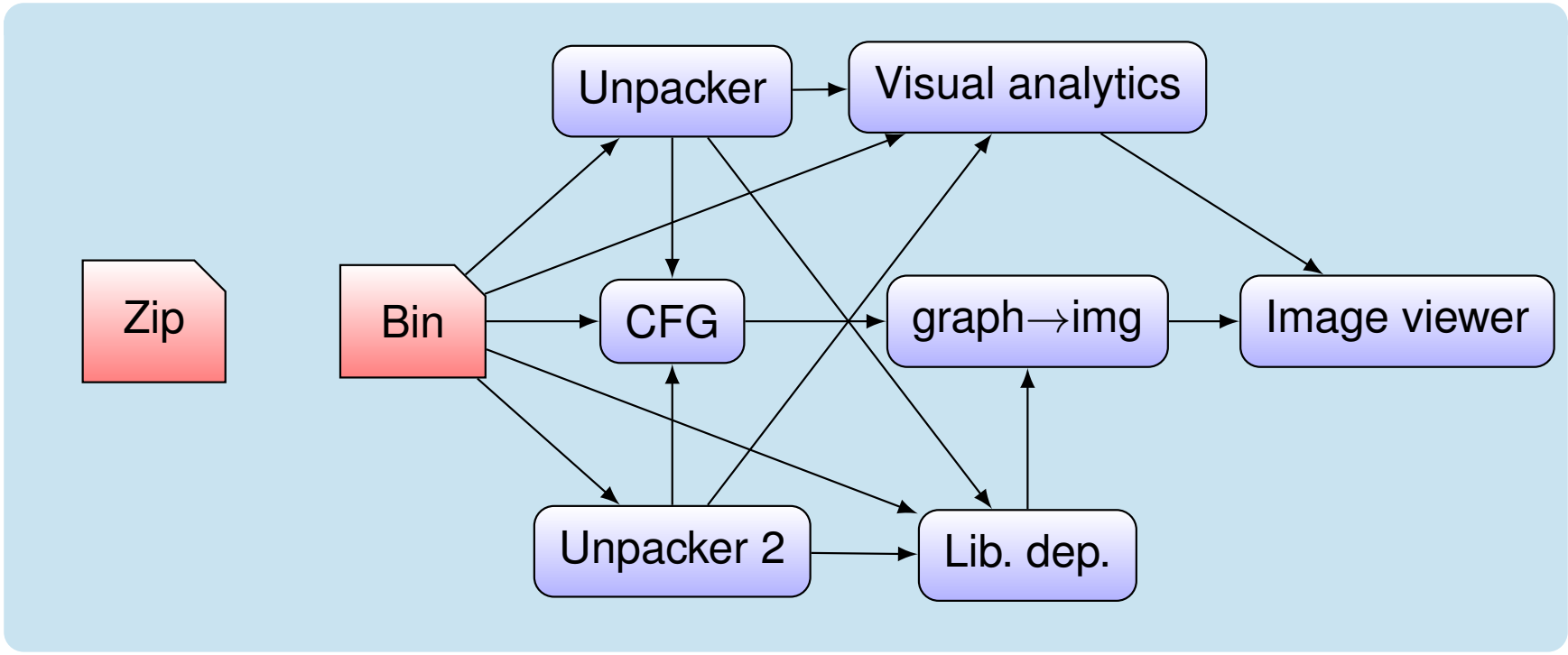
Exemple de workflow d'analyse



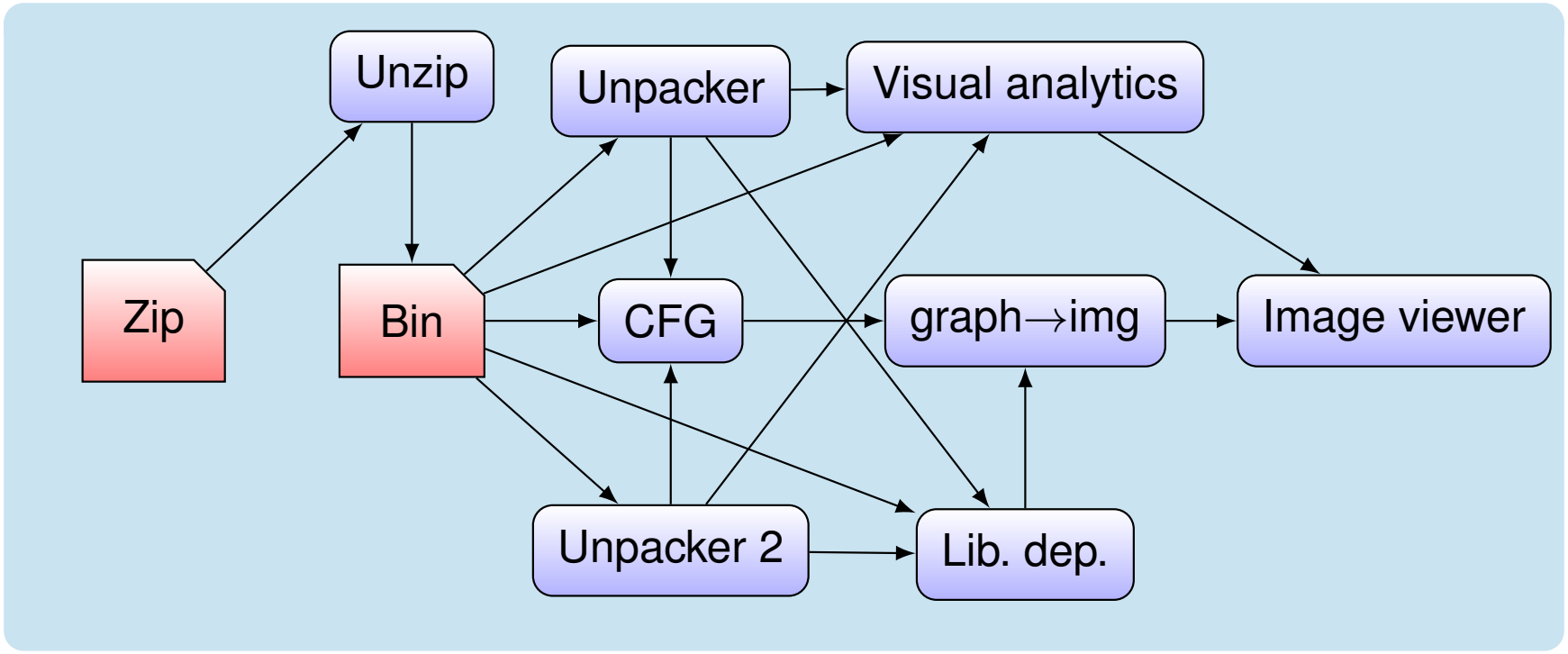
Exemple de workflow d'analyse



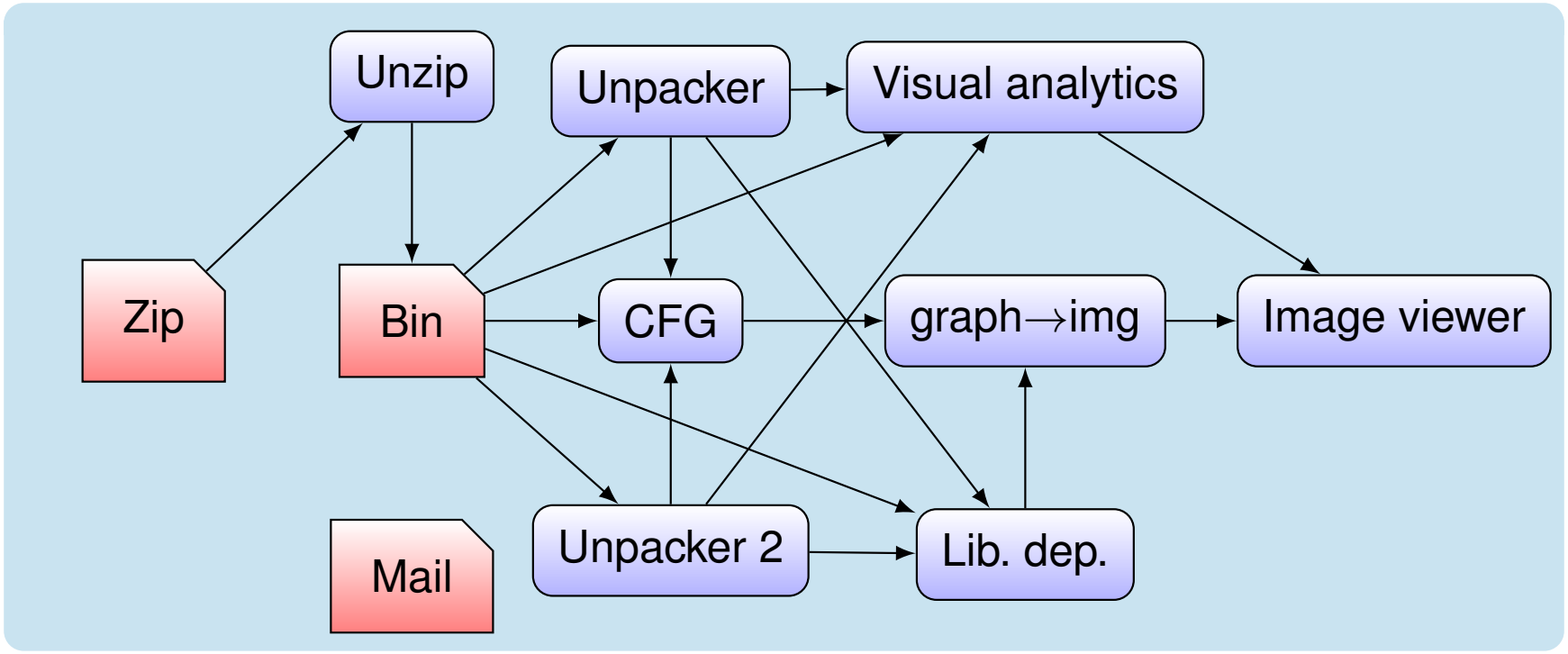
Exemple de workflow d'analyse



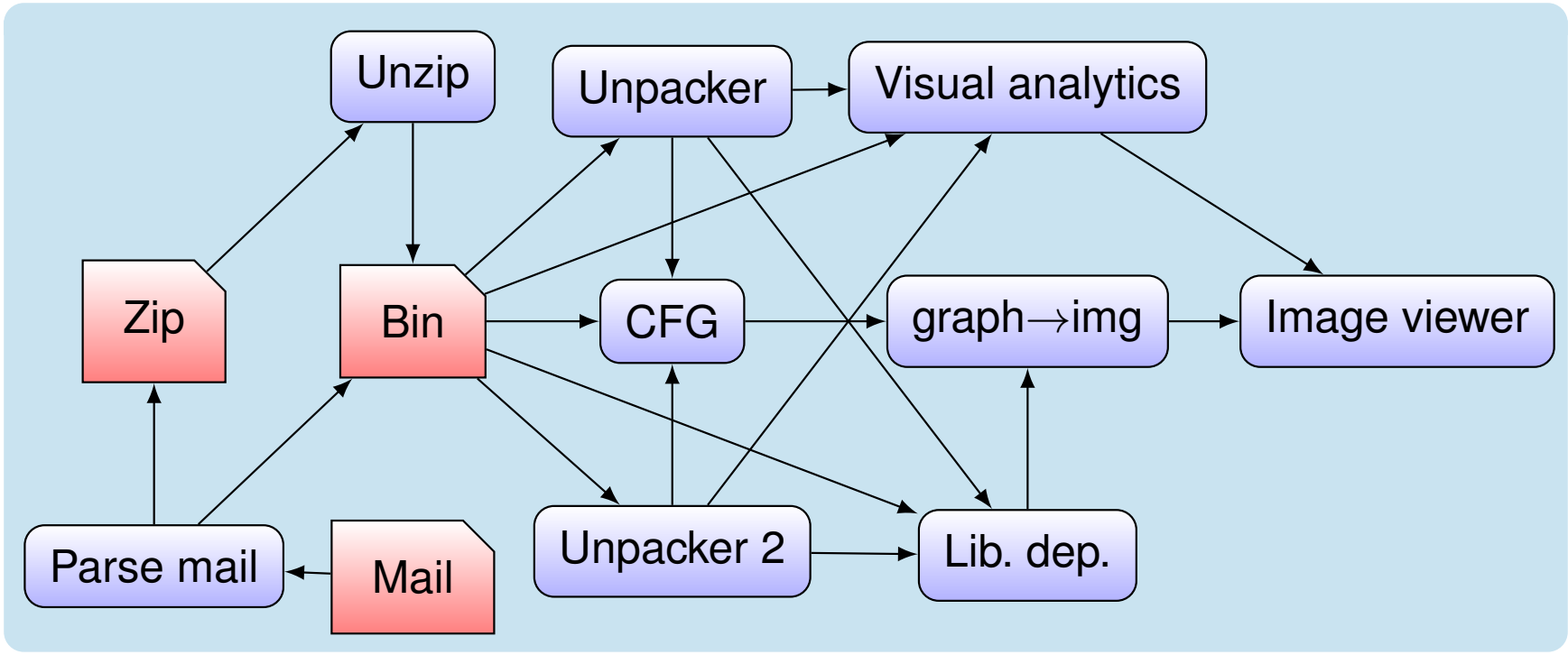
Exemple de workflow d'analyse

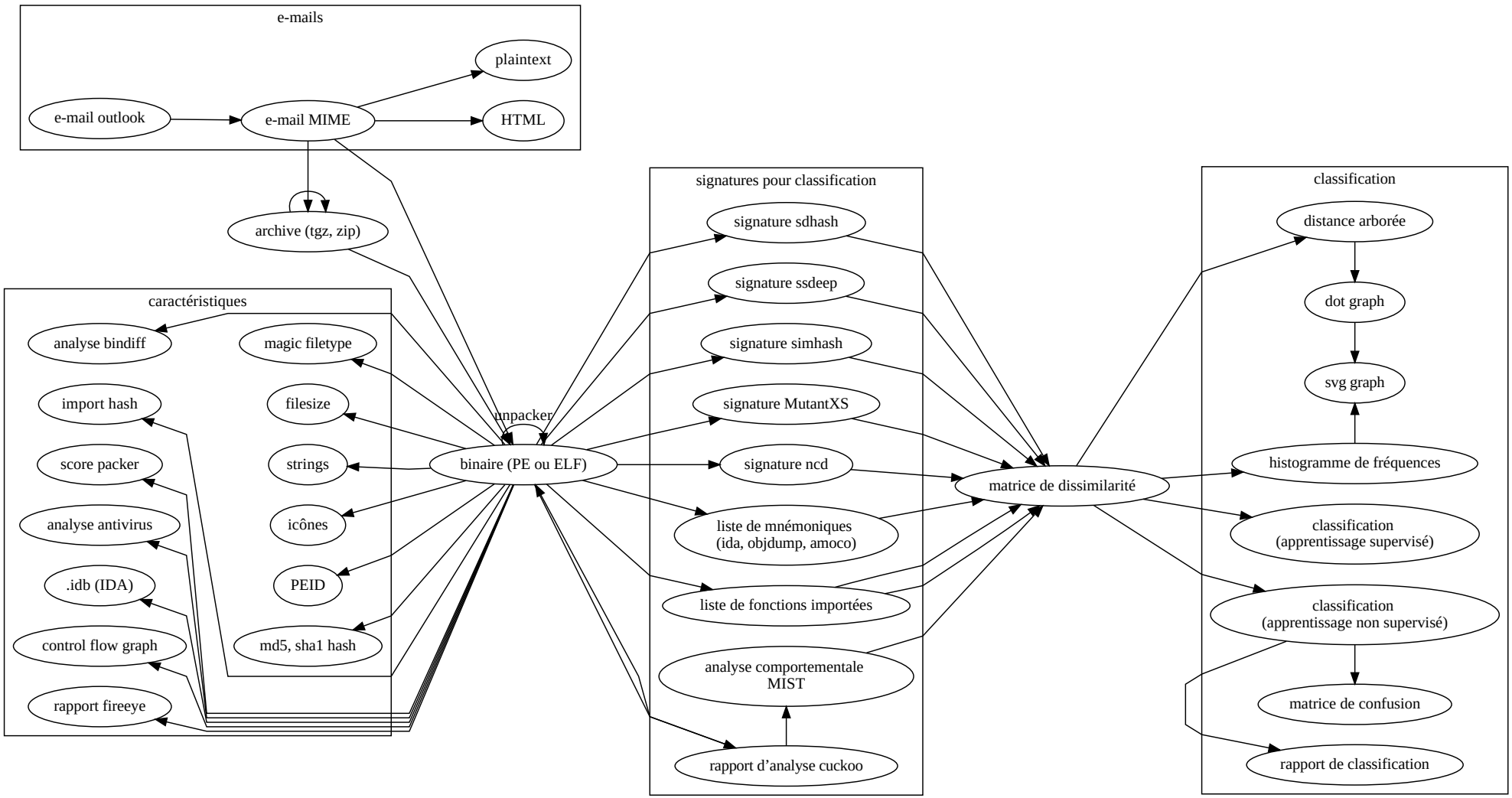


Exemple de workflow d'analyse



Exemple de workflow d'analyse




e-mails
plaintext
e-mail outlook
e-mail MIME
HTML
archive (tgz, zip)
caractéristiques
analyse bindiff
magic filetype
import hash
filesize
score packer
strings
analyse antivirus
icônes
.idb (IDA)
PEID
control flow graph
md5, sha1 hash
rapport fireeye
unpacker
binaire (PE ou ELF)
signatures pour classification
signature sdhash
signature ssdeep
signature simhash
signature MutantXS
signature ncd
liste de mnémoniques
(ida, objdump, amoco)
liste de fonctions importées
analyse comportementale
MIST
rapport d'analyse cuckoo
matrice de dissimilarité
classification
distance arborée
dot graph
svg graph
histogramme de fréquences
classification
(apprentissage supervisé)
classification
(apprentissage non supervisé)
matrice de confusion
rapport de classification

Objectifs de REbus

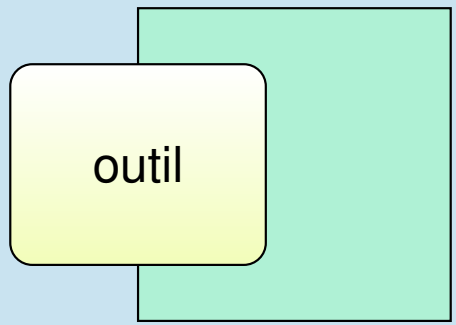
Faciliter le travail de l'outilleur et de l'analyste

- Ajout aisé d'une nouvelle fonctionnalité présente dans un outil externe
- Ajout aisé d'une variante d'une fonctionnalité déjà présente
- Utilisation comme « super-outil » ou infrastructure complète d'analyse
- Automatisation des tâches répétitives
- Passage à l'échelle
- Reproductibilité des analyses
- Stockage des résultats intermédiaires

Considérations architecturales

Faire sur mesure

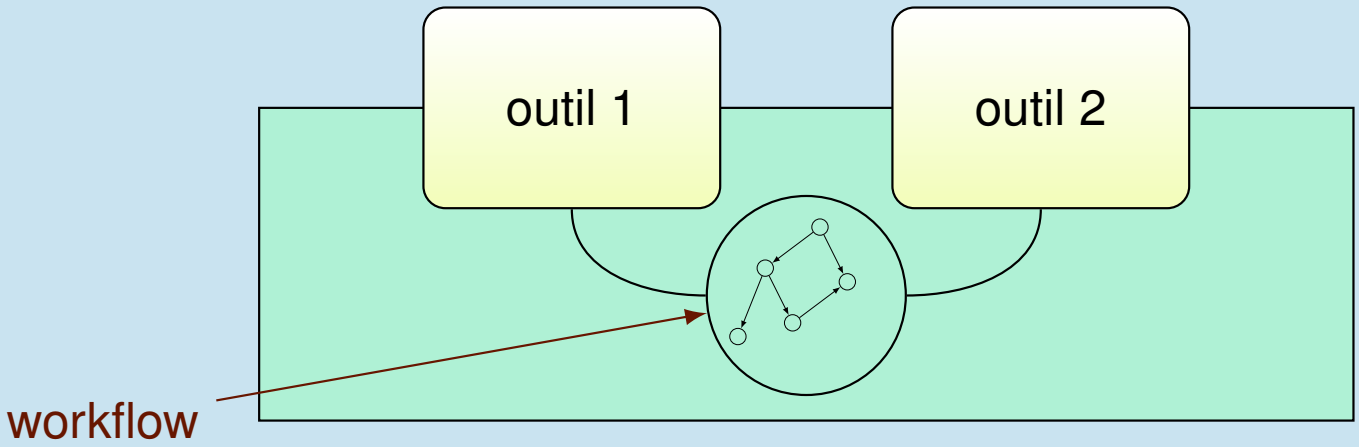
Wrapper un outil



Considérations architecturales

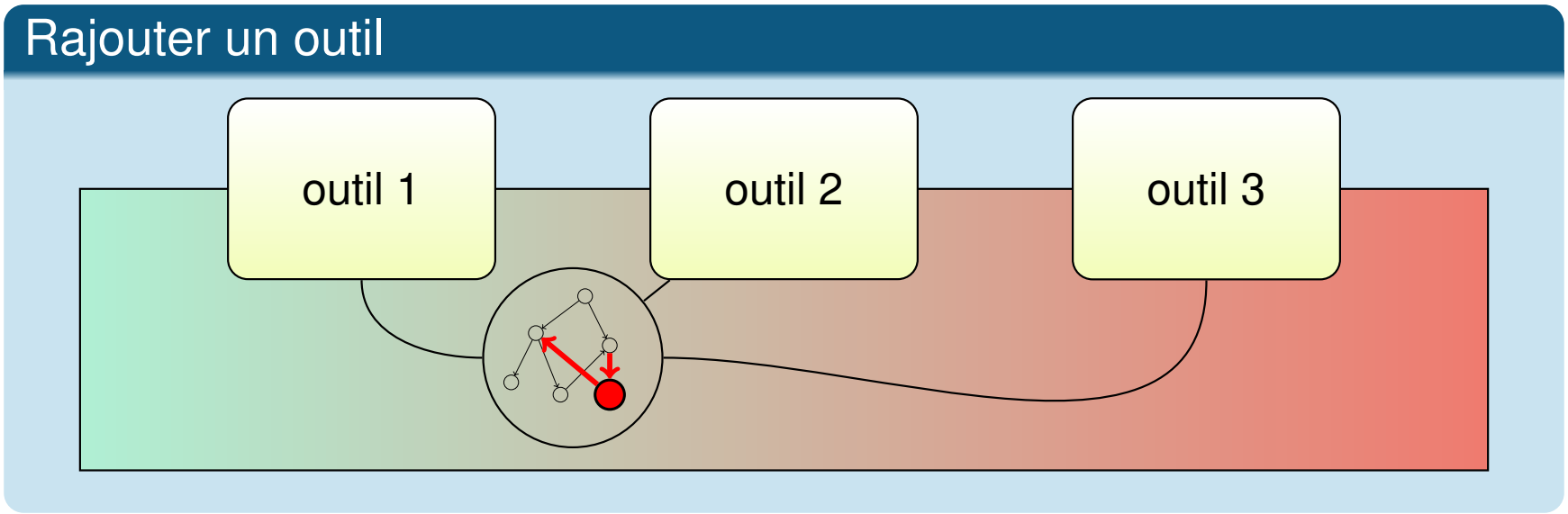
Faire sur mesure

Faire communiquer plusieurs outils



Considérations architecturales

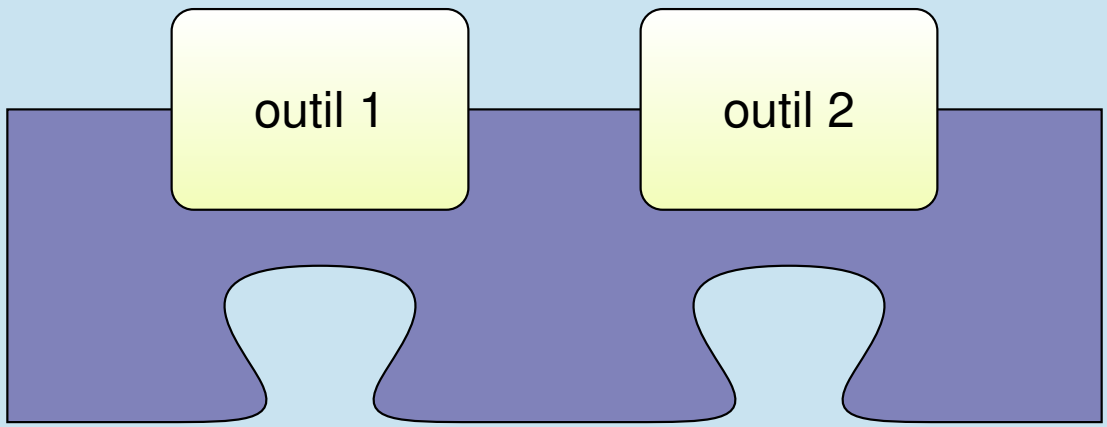
Faire sur mesure



Considérations architecturales

Bibliothèques

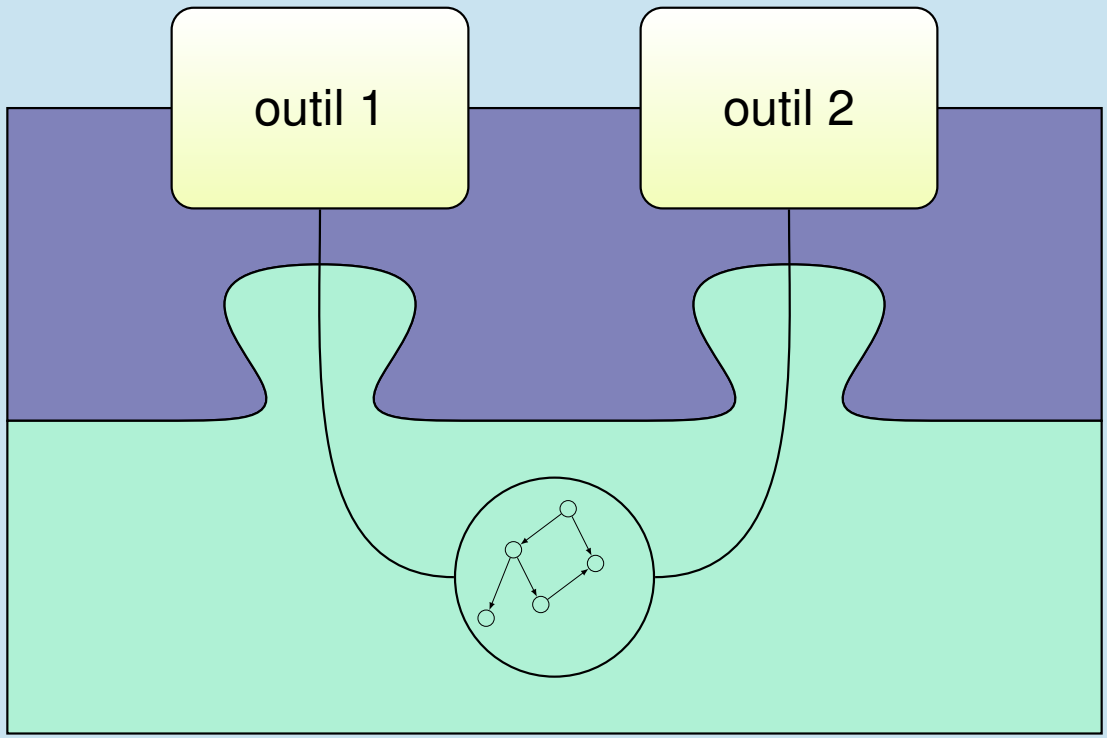
Faire une lib de wrappers



Considérations architecturales

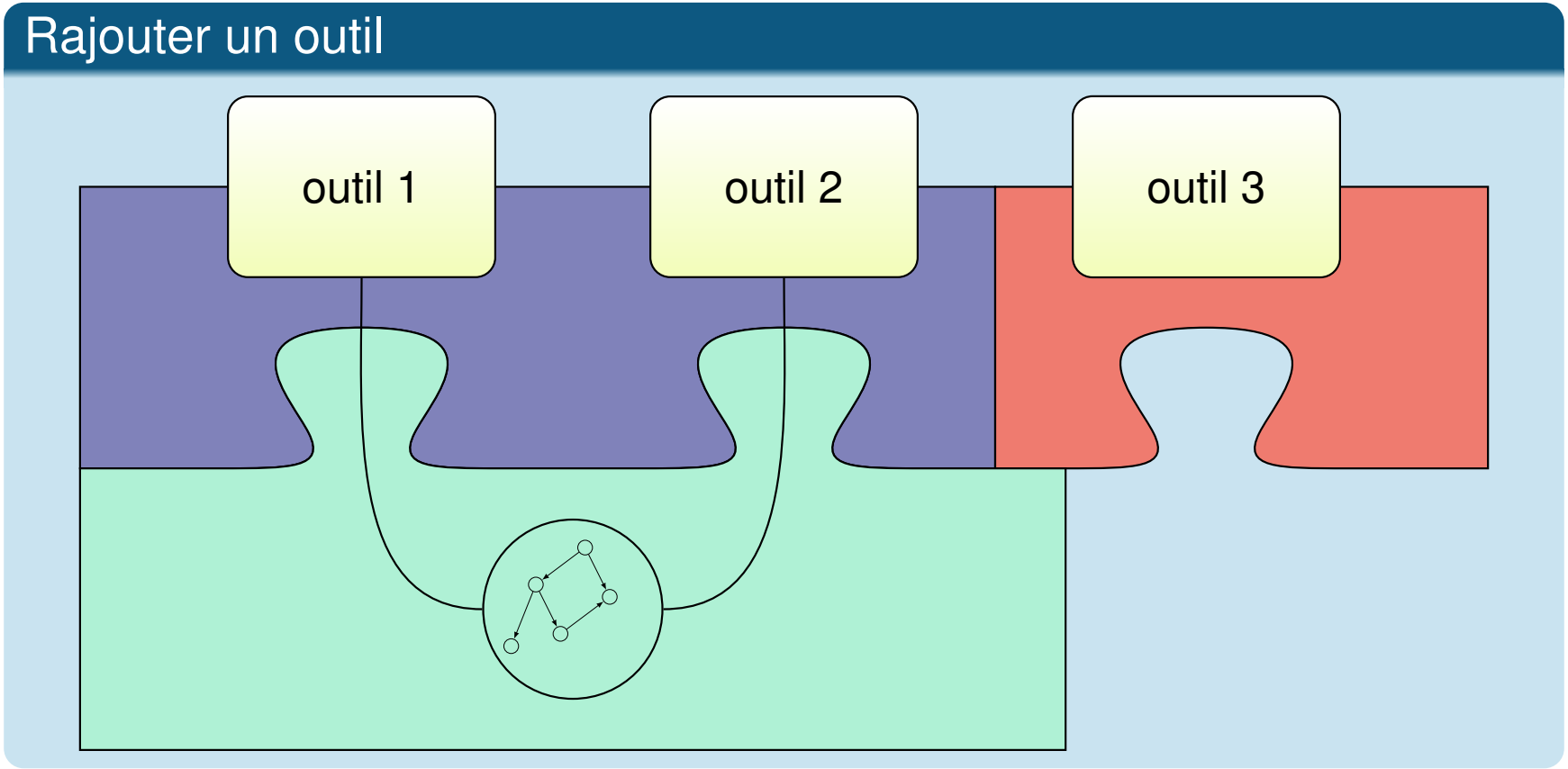
Bibliothèques

Utiliser la bibliothèque



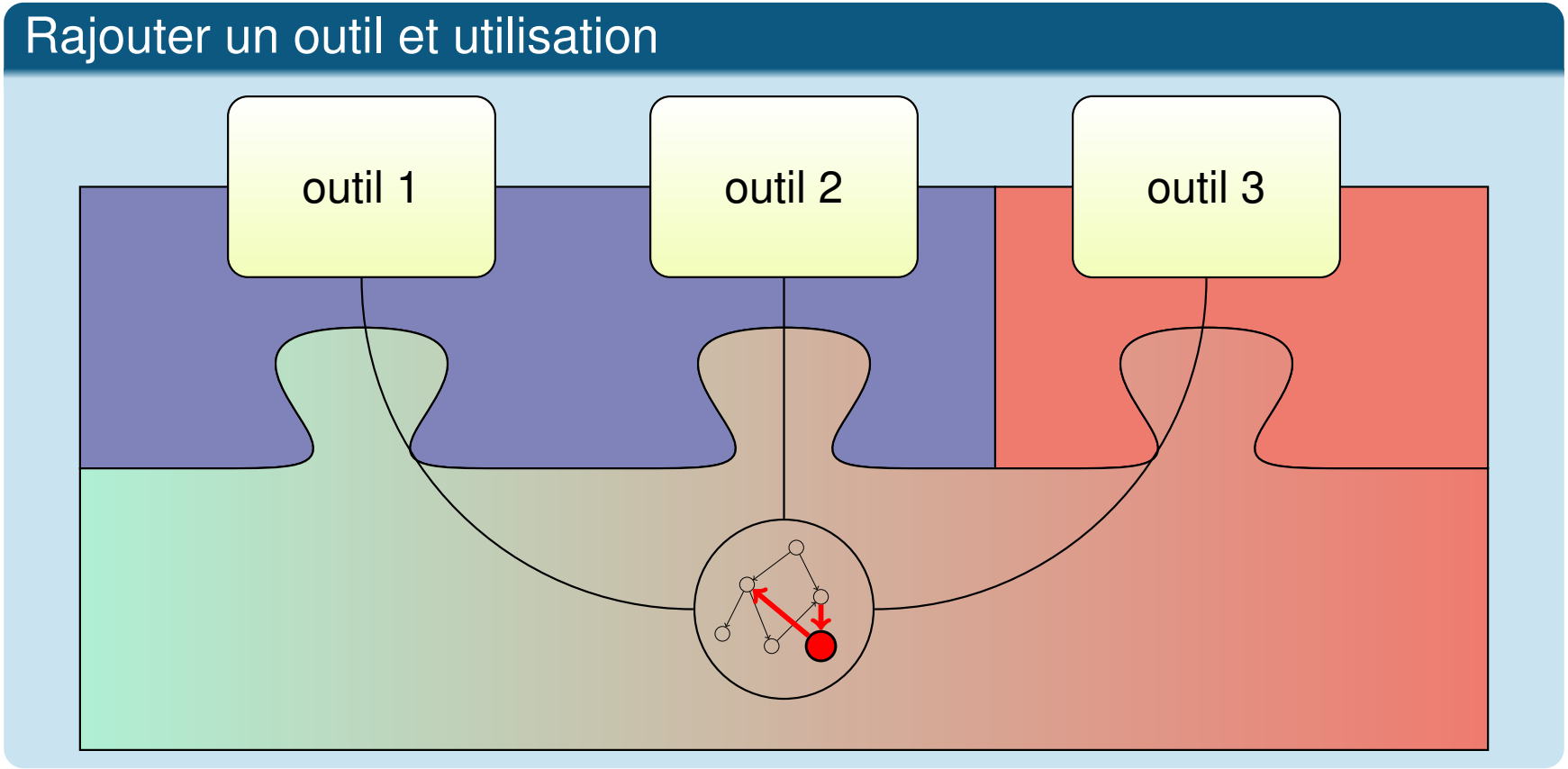
Considérations architecturales

Bibliothèques



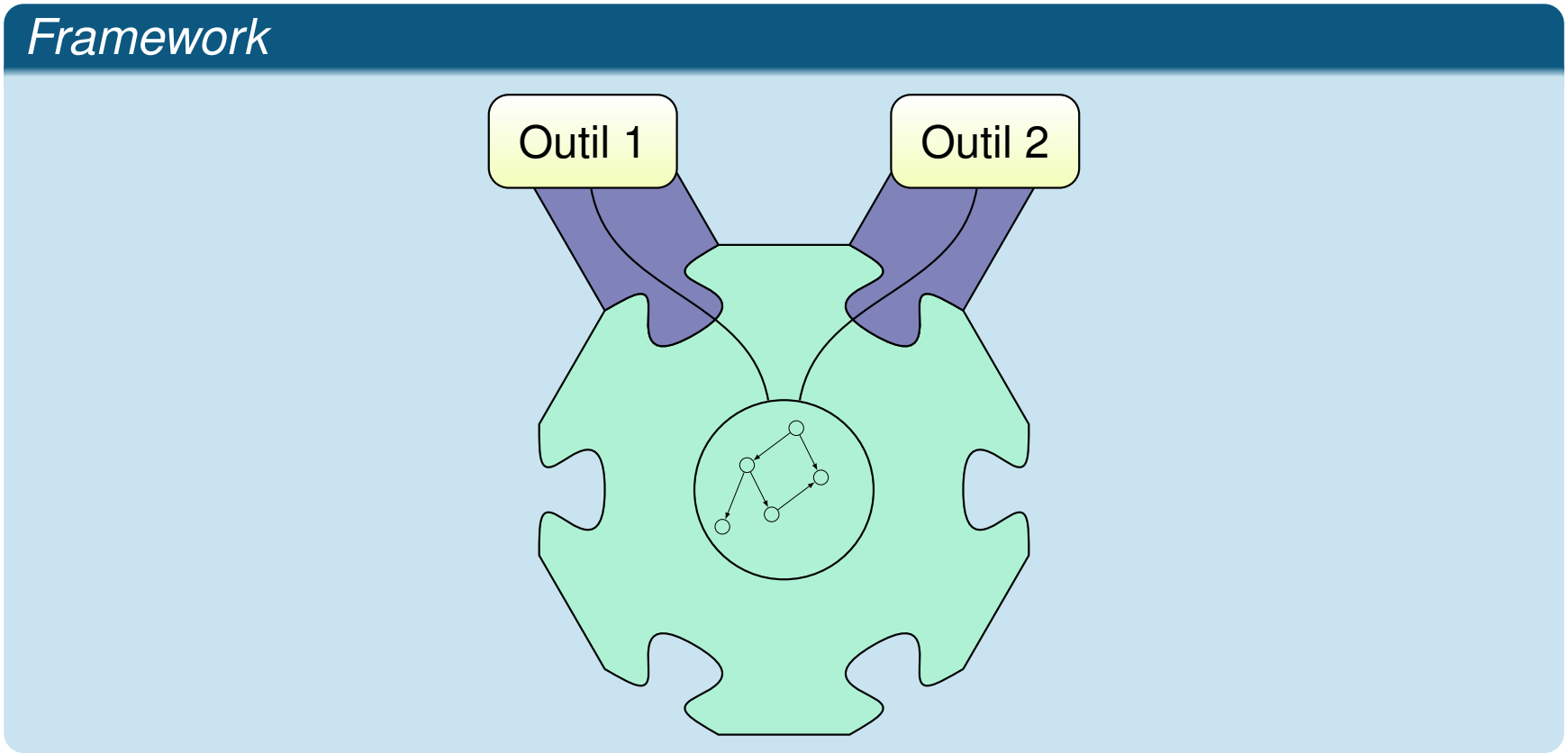
Considérations architecturales

Bibliothèques



Considérations architecturales

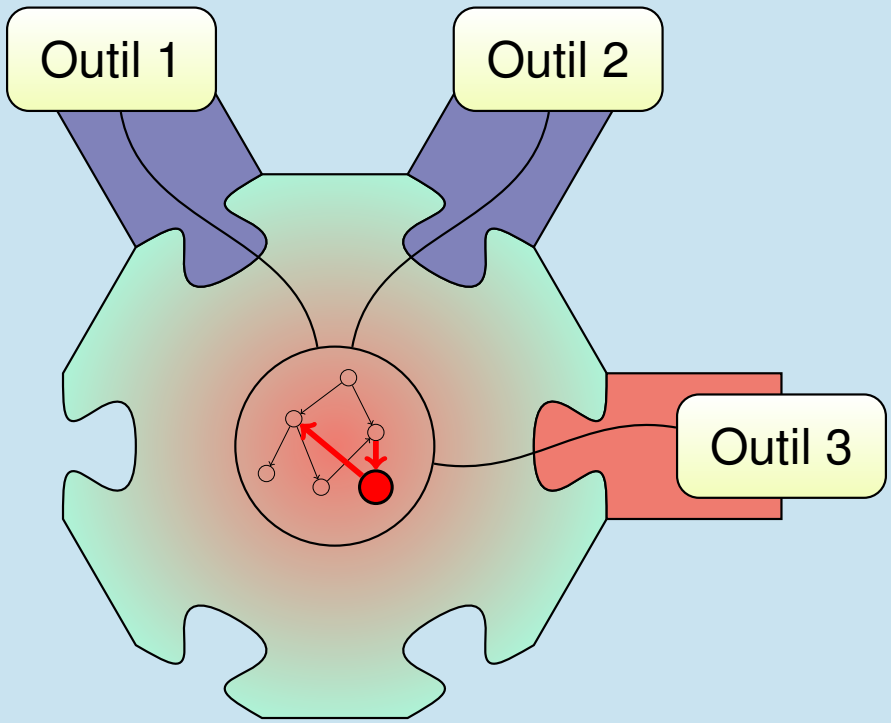
Approche de type *framework*, workflow centralisé



Considérations architecturales

Approche de type *framework*, *workflow* centralisé

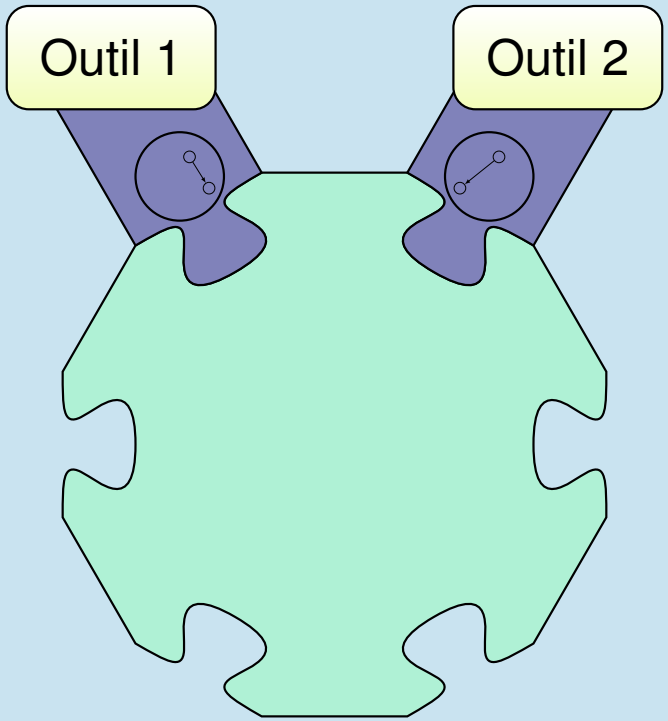
Rajout d'un *plug-in*



Considérations architecturales

Approche de type *framework*, *workflow* décentralisé

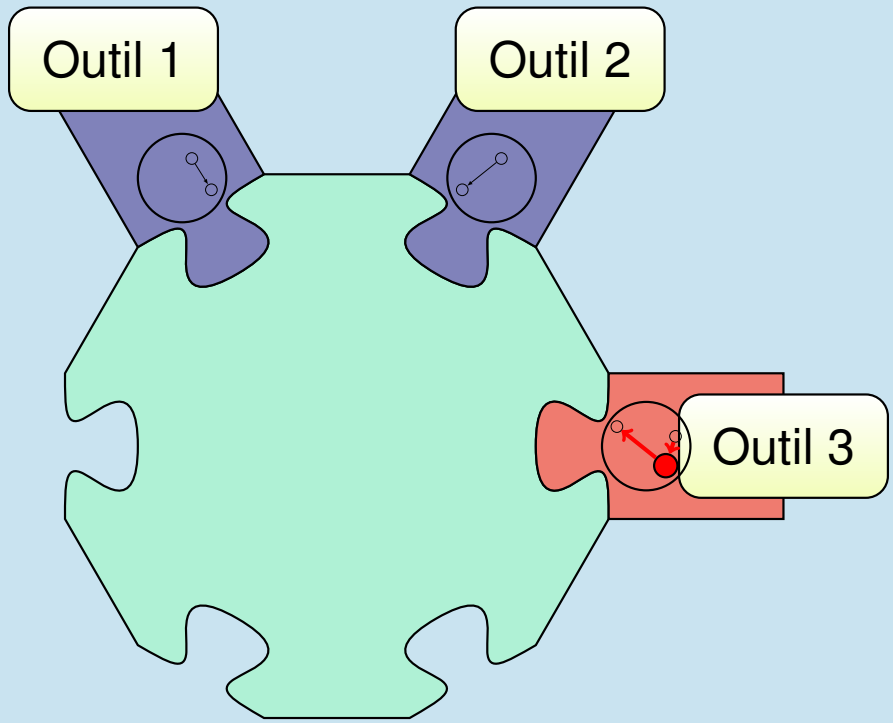
Workflow décentralisé



Considérations architecturales

Approche de type *framework*, workflow décentralisé

Ajout d'un plug-in



Plan

Plan

Choix de conception

- Framework avec workflow décentralisé
- Sous la forme d'un bus de communication
- Un composant centralisé (*bus master*) pour faire circuler les messages
- La décision de traiter une donnée revient à l'agent (workflow décentralisé)
- Indépendant du mécanisme de transport sous-jacent (ex : DBus)
- Exhaustivité privilégiée à l'optimisation : des traitements superflus seront peut-être effectués

Sécurité de REbus

Pas un objectif (pour l'instant)

- Pas d'authentification, contrôle d'accès, contrôle d'intégrité
- Pas encore d'isolation automatique des outils d'analyse potentiellement exploitables par les données analysées

$\Longrightarrow$ pour l'instant, traiter les données potentiellement dangereuses dans un environnement isolé

Concepts de REbus

Composants de REbus

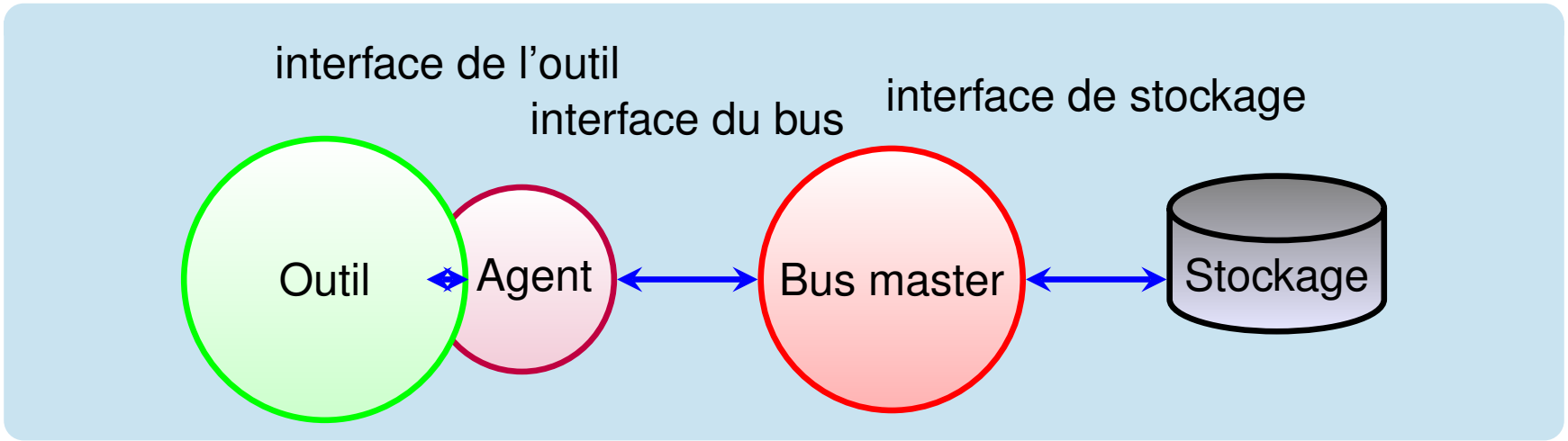
Descripteur : contient la donnée et ses métadonnées associées

Agent : pilote un outil externe, interface entre ses entrées-sorties et le bus

Bus : permet la communication entre agents (transport)

Bus Master : reçoit tous les messages, s'assure du stockage, répond aux requêtes des agents

Interfaces de REbus

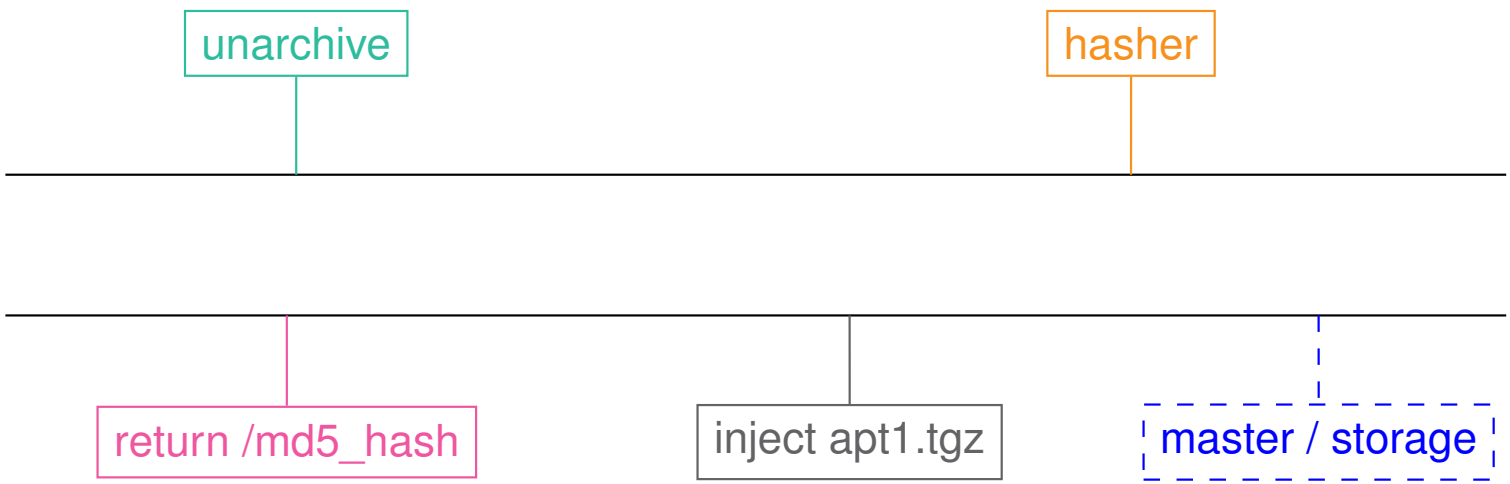


Dynamique des échanges entre agents sur le bus

Description de l'outil obtenu

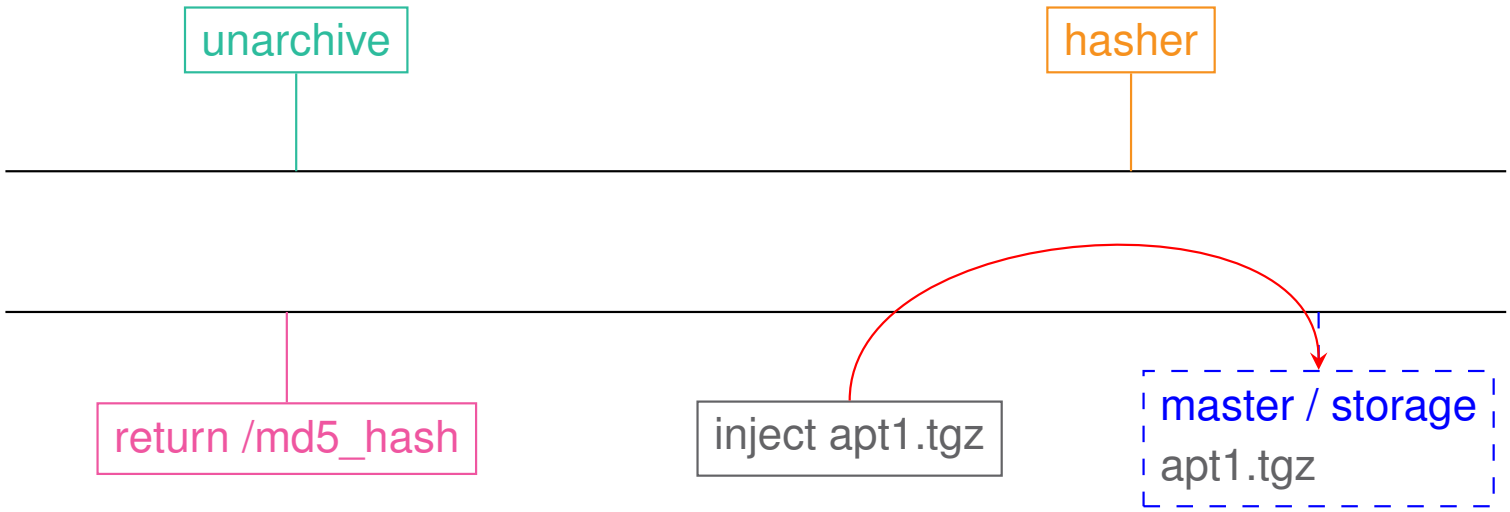
- extrait des fichiers `.tgz`
- calcule le *hash* MD5 de chaque fichier contenu dans l'archive
- affiche le *hash* MD5 sur la sortie standard

```
$ rebus_agent -m rebus_demo.agents hasher unarchive \
                                   inject ~/apt1.tgz -- \
                                   return --short md5_hash
```



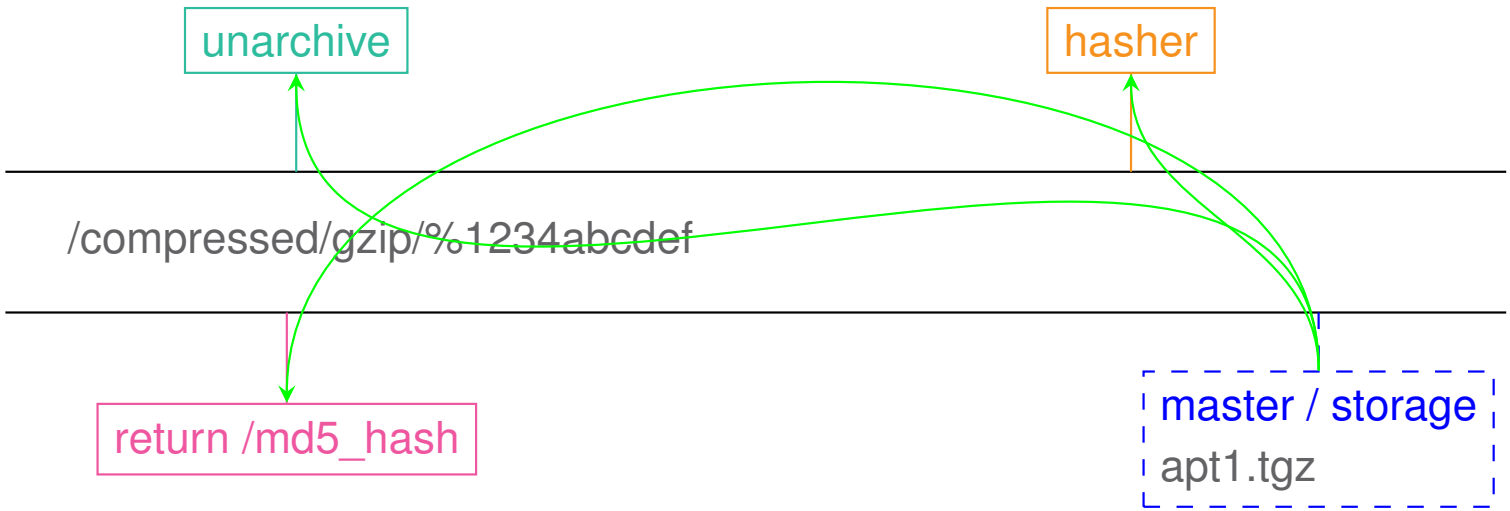
Dynamique des échanges entre agents sur le bus

```
$ rebus_agent -m rebus_demo.agents hasher unarchive \
                                   inject ~/apt1.tgz -- \
                                   return --short md5_hash
```



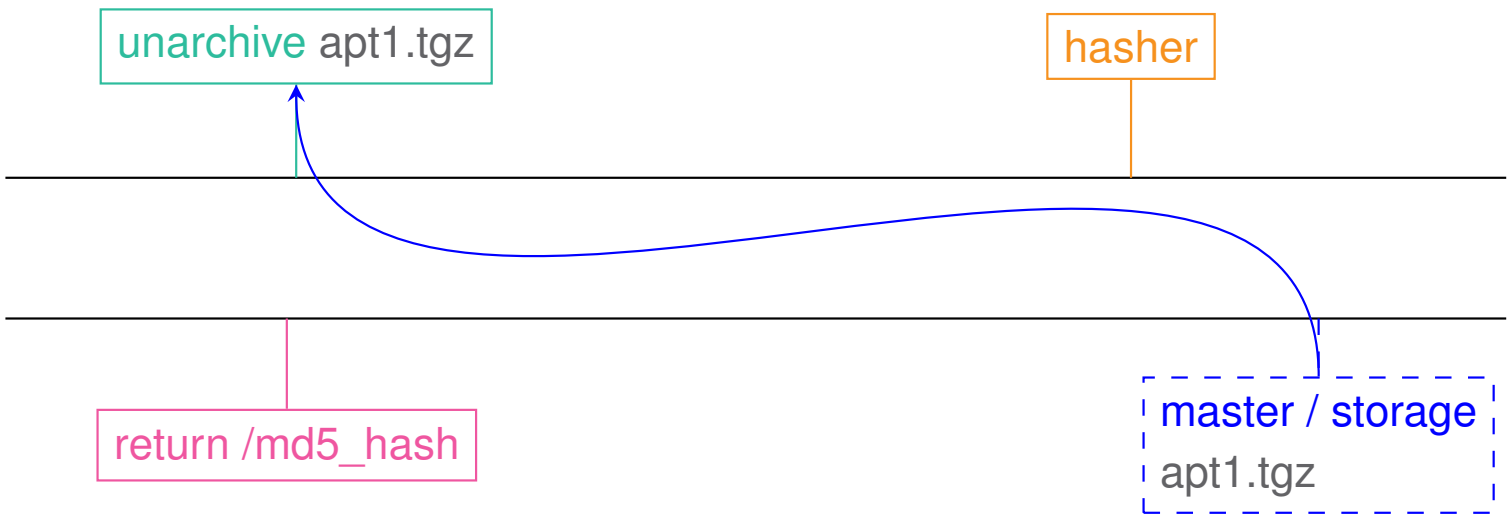
Dynamique des échanges entre agents sur le bus

```
$ rebus_agent -m rebus_demo.agents hasher unarchive \
                                   inject ~/apt1.tgz -- \
                                   return --short md5_hash
```



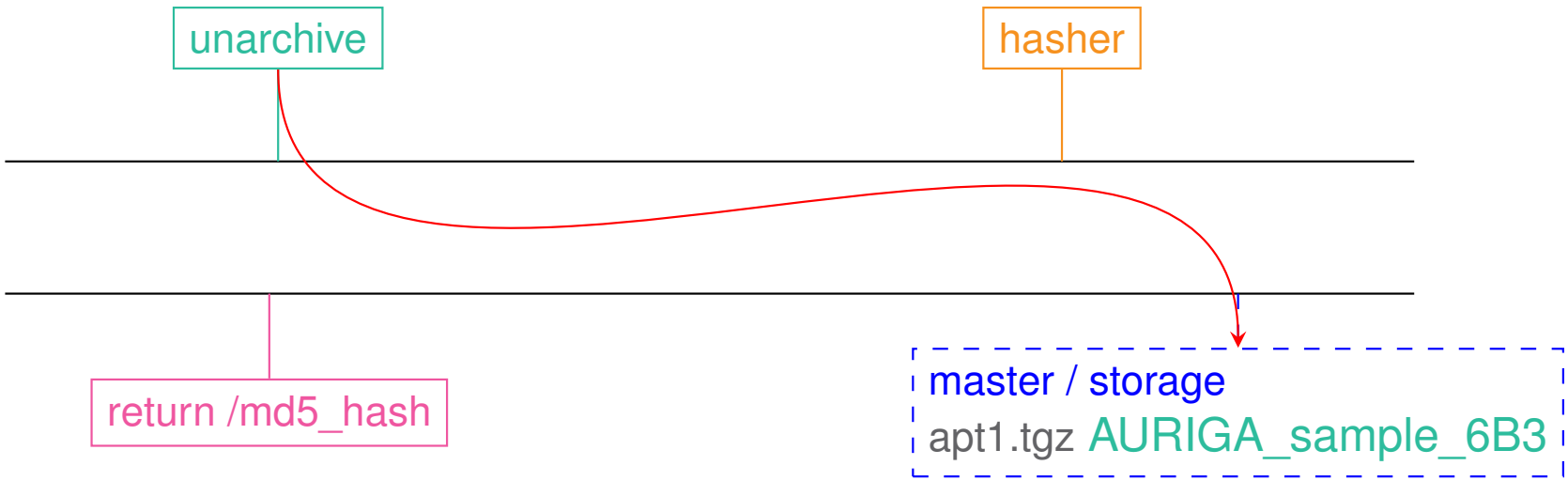
Dynamique des échanges entre agents sur le bus

```
$ rebus_agent -m rebus_demo.agents hasher unarchive \
                                   inject ~/apt1.tgz -- \
                                   return --short md5_hash
```



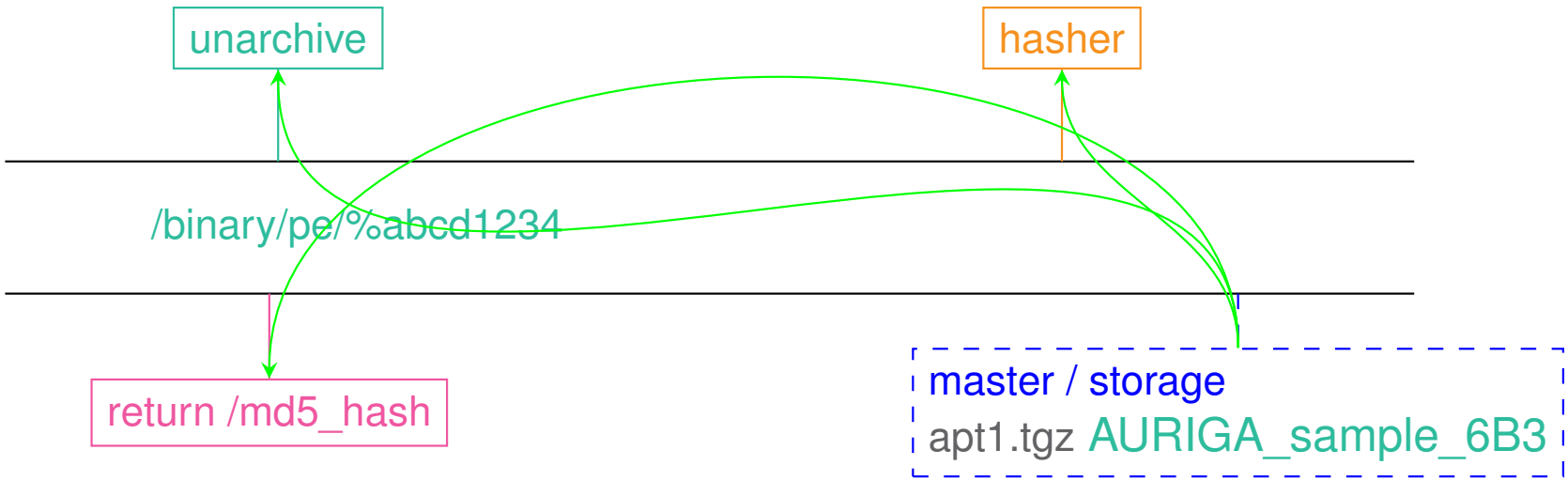
Dynamique des échanges entre agents sur le bus

```
$ rebus_agent -m rebus_demo.agents hasher unarchive \
                                   inject ~/apt1.tgz -- \
                                   return --short md5_hash
```



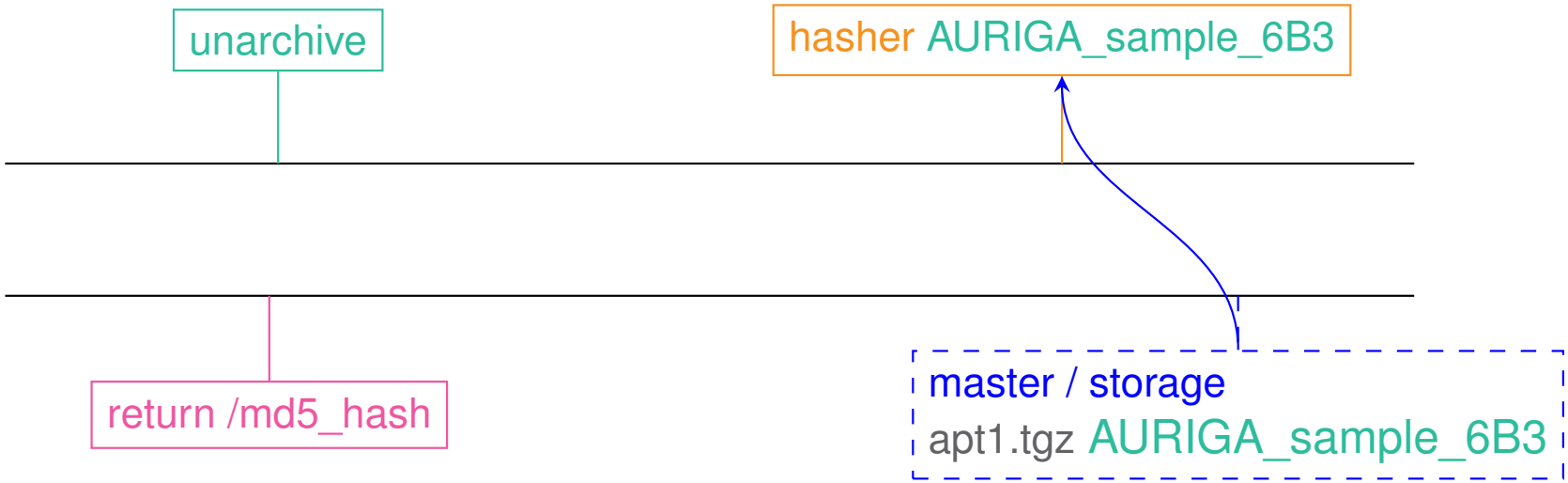
Dynamique des échanges entre agents sur le bus

```
$ rebus_agent -m rebus_demo.agents hasher unarchive \
                                   inject ~/apt1.tgz -- \
                                   return --short md5_hash
```



Dynamique des échanges entre agents sur le bus

```
$ rebus_agent -m rebus_demo.agents hasher unarchive \
                                   inject ~/apt1.tgz -- \
                                   return --short md5_hash
```

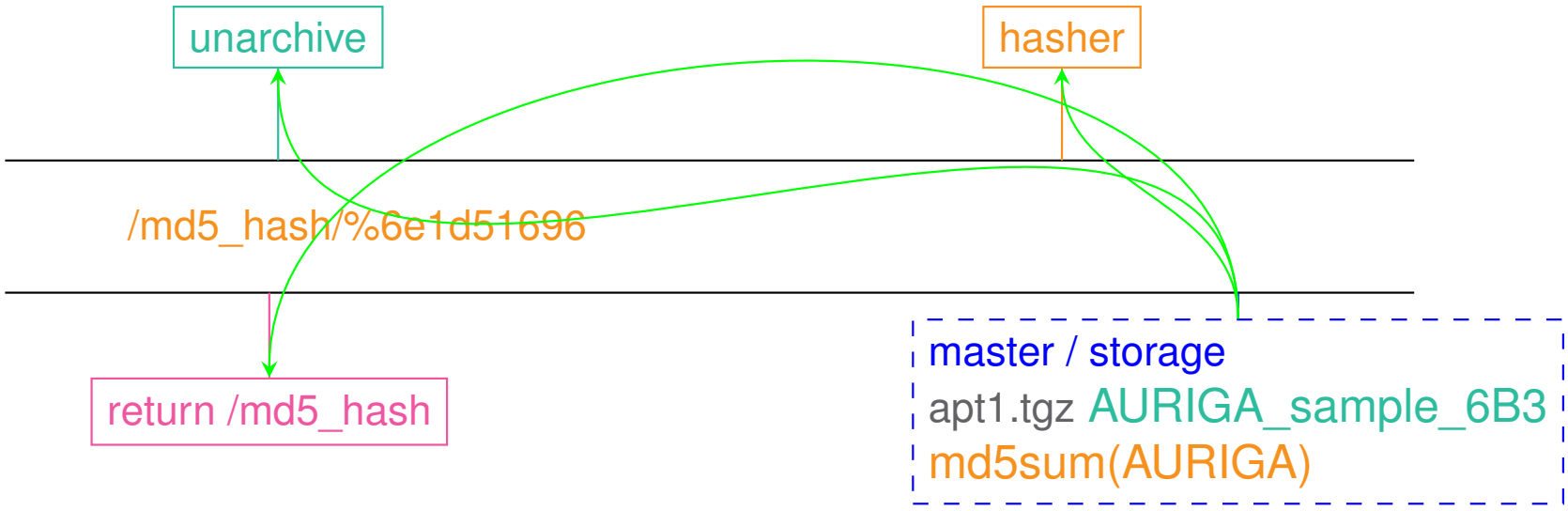


Dynamique des échanges entre agents sur le bus

```
$ rebus_agent -m rebus_demo.agents hasher unarchive \
                                   inject ~/apt1.tgz -- \
                                   return --short md5_hash
```

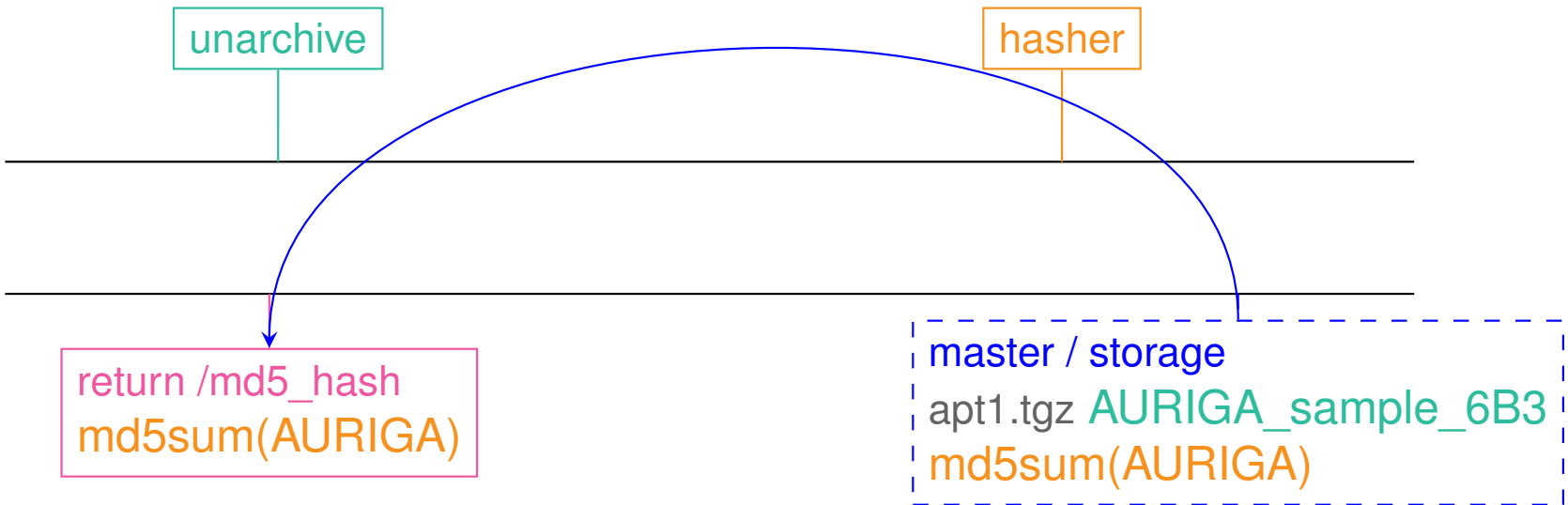
Dynamique des échanges entre agents sur le bus

```
$ rebus_agent -m rebus_demo.agents hasher unarchive \
                                   inject ~/apt1.tgz -- \
                                   return --short md5_hash
```



Dynamique des échanges entre agents sur le bus

```
$ rebus_agent -m rebus_demo.agents hasher unarchive \
                                   inject ~/apt1.tgz -- \
                                   return --short md5_hash
```

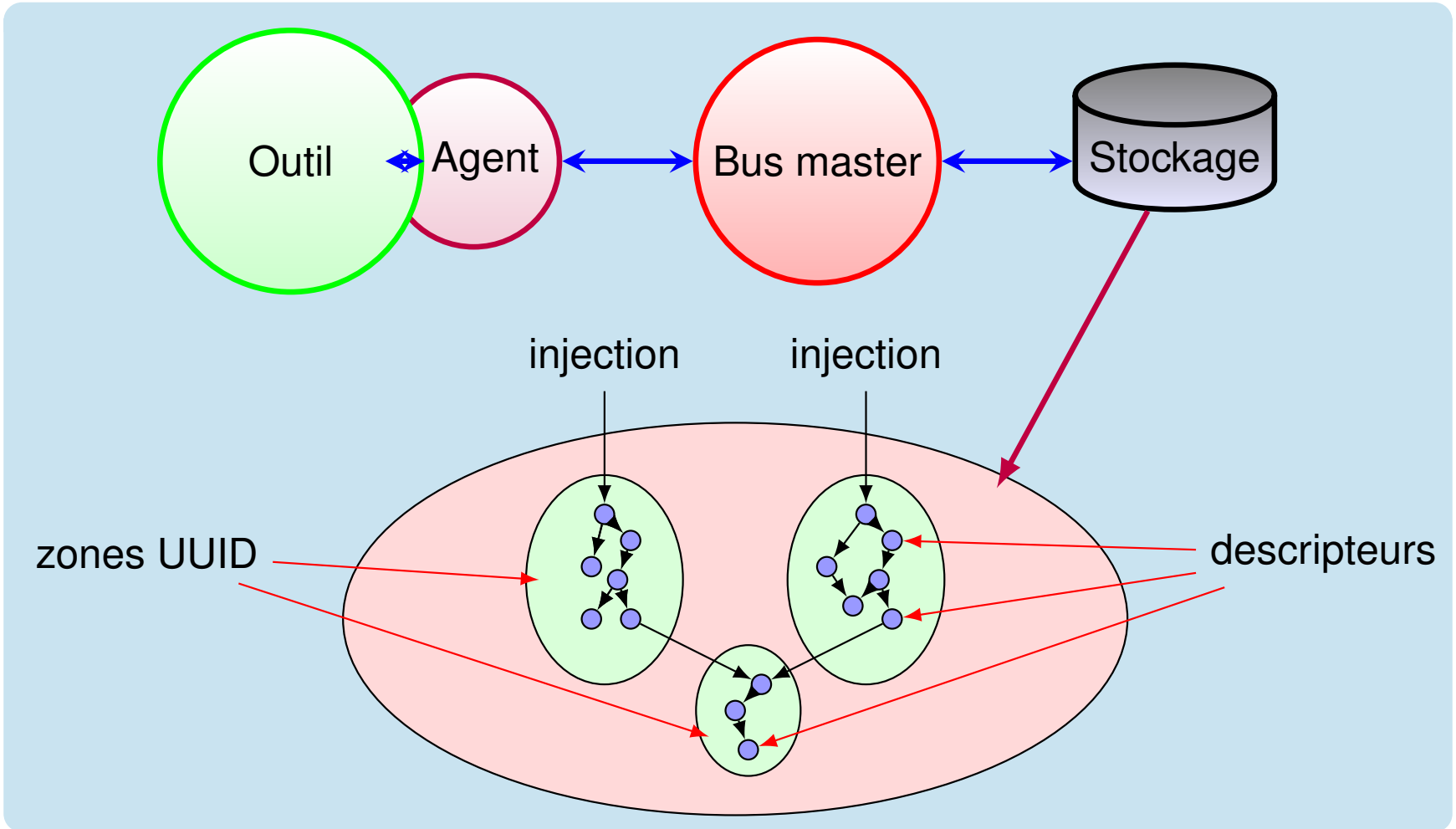


LocalBus - exemple de combinaison d'agents

```
$ rebus_agent -m rebus_demo.agents hasher unarchive \
                                   inject ~/apt1.tgz -- \
                                   return --short md5_hash

apt1.tgz:AURIGA_6B31344B40E2AF9C9EE3BA707558C14E =
    6b31344b40e2af9c9ee3ba707558c14e
apt1.tgz:AURIGA_CDCD3A09EE99CFF9A58EFEA5CCBE2BED =
    cdcd3a09ee99cff9a58efea5ccbe2bed
apt1.tgz:BANGAT_468FF2C12CFFC7E5B2FE0EE6BB3B239E =
    468ff2c12cffc7e5b2fe0ee6bb3b239e
[...]
```

La soupe de descripteurs



Plan

```python
from rebus.agent import Agent
import hashlib

@Agent.register
class Hasher(Agent):
    _name_ = "hasher"
    _desc_ = "Return md5 of a binary"

    def selector_filter(self, selector):
        # Cet agent traite uniquement les descripteurs dont le
        # sélecteur commence par "/binary/"
        return selector.startswith("/binary/")

    def process(self, desc, sender_id):
        # Compute md5 hash value
        md5_hash = hashlib.md5(desc.value).hexdigest()

        # Create a new child descriptor
        new_desc = desc.spawn_descriptor(
            "/md5_hash", unicode(md5_hash), self.name)

        # Push the new descriptor to the bus
        self.push(new_desc)
```

Listing 1 – Agent REbus calculant le hash MD5 de fichiers binaires

Agent

```
class Hasher(Agent):
    _name_ = "hasher"
    _desc_ = "Return md5 of a binary"
```

Agent

- Pilote un outil externe
- Interface avec le bus :
  - Choix des données traitées
  - Transformations des données
  - Envoi des données traitées

Filtrage des entrées

```
def selector_filter(self, selector):
    # Cet agent traite uniquement les descripteurs dont le
    # selecteur commence par "/binary/"
    return selector.startswith("/binary/")
```

Sélecteur

- Exemple :
  `/signature/md5/%6e1d5169661a50(...)f989129a583f92b9dee`
- Décrit le type de la données encapsulée ($\simeq$ type MIME) : image, fichier binaire, etc.
- Décrit le format de la donnée (ex. JPG ou PNG pour une image, PE ou ELF pour un binaire, etc.),
- Identifié de manière unique par un `hash` (SHA256)

Traitement des données

```
def process(self, desc, sender_id):
    # Compute md5 hash value
    md5_hash = hashlib.md5(desc.value).hexdigest()
```

Exemple de descripteur

```
{'selector': '/binary/pe/%aaf3ad(...)3d50aad60',
'_value': 'MZ\x90\x00\x03\x00\x00\x00\x04\x00\x00\x00...'
'label': 'AURIGA_sample_6B31344B40E2AF9C9EE3BA707558C14E',
'agent': (reference vers l'objet agent),
'bus': (reference vers l'objet bus),
'domain': 'default',
'hash': '403f73b357(...)8841100871b',
'precursors': [],
'processing_time': 0.0135,
'uuid': 'c0cd0811-f108-5447-8bc0-eec2b53311d9',
'version': 0}
```

Création et envoi d'un descripteur

```
def process(self, desc, sender_id):
    # Compute md5 hash value
    md5_hash = hashlib.md5(desc.value).hexdigest()

    # Create a new child descriptor
    new_desc = desc.spawn_descriptor(
        "/md5_hash", unicode(md5_hash), self.name)

    # Push the new descriptor to the bus
    self.push(new_desc)
```

Création d'un descripteur

- Choix du sélecteur
- Valeur
- Envoi vers bus

Modes de traitement des descripteurs

Trois modes de fonctionnement possibles

- Automatique
- Sur demande de l'utilisateur
- Lorsque le bus est inactif

Plan

Implémentations de l'API bus

Les agents sont indépendants de l'implémentation du bus

Implémentations existantes

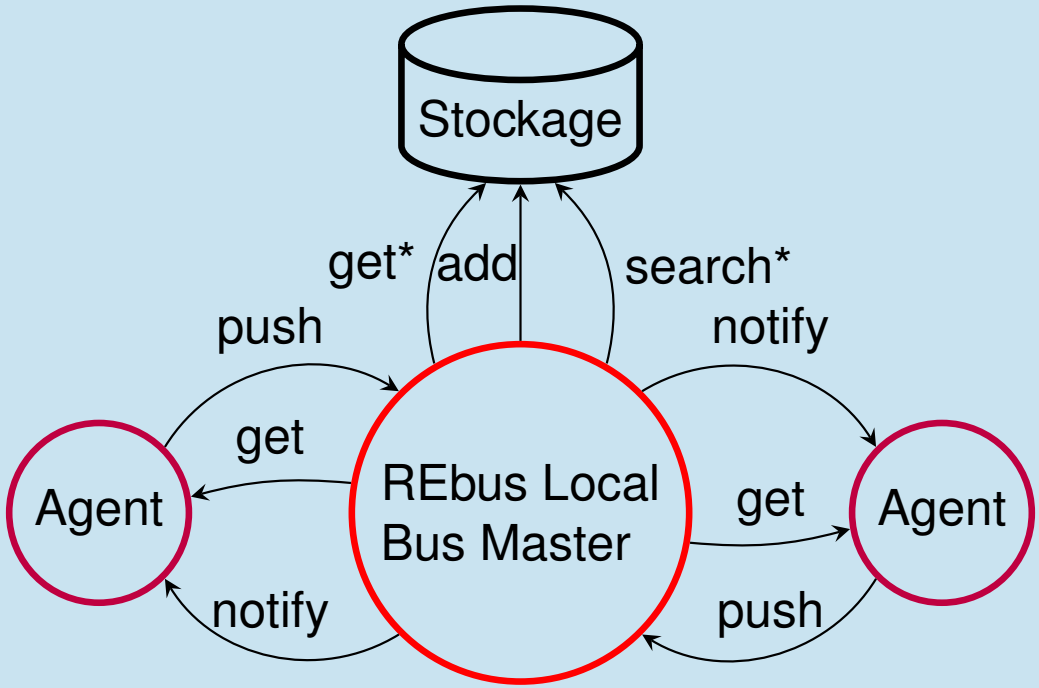
- localbus, mode super-outil
- REbus over DBus, mode interactif

Implémentations prévues

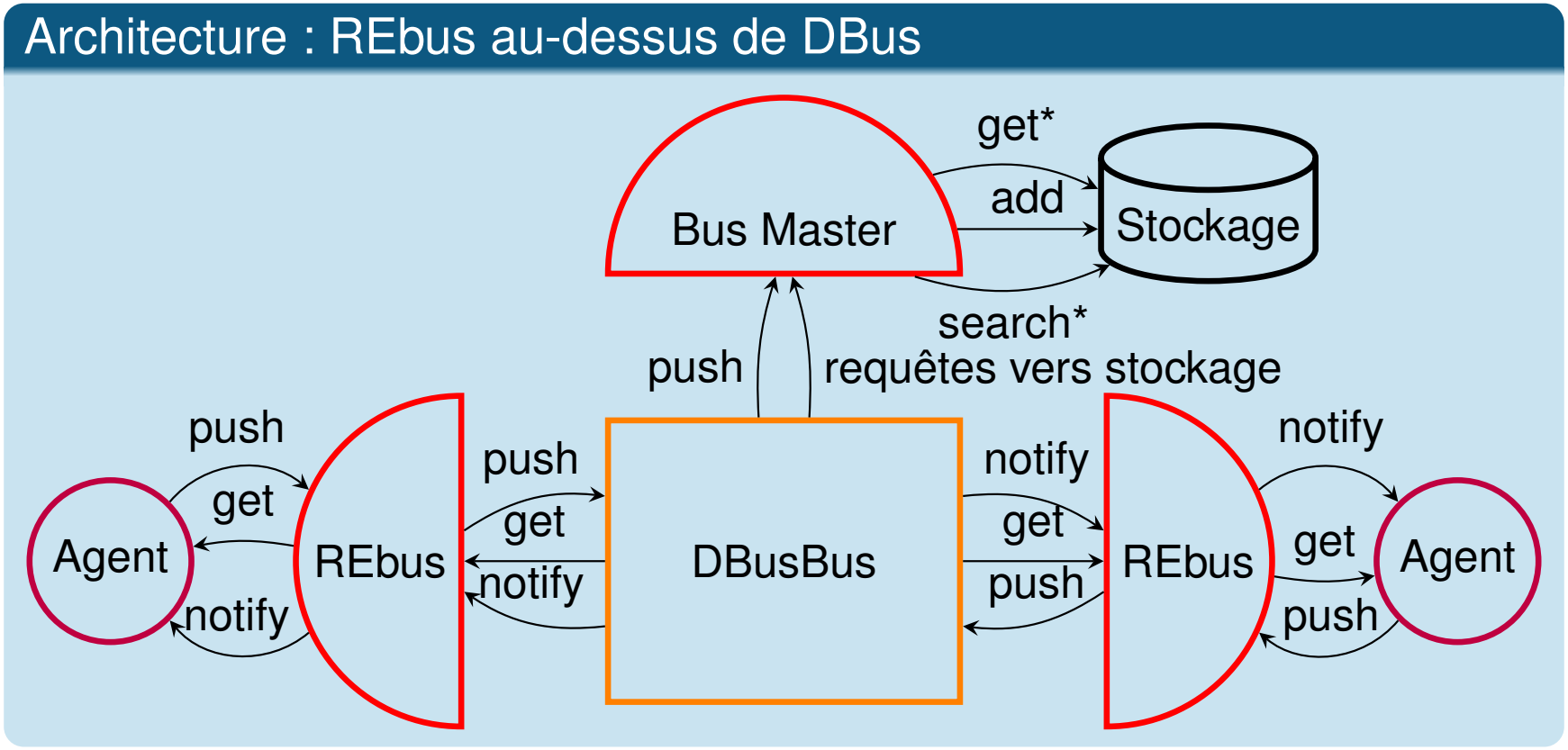
- MPI
- HTTP REST
- 0mq

Implémentation de l'API Bus : Localbus

Architecture : REbus Local Bus



Implémentation de l'API Bus : DBus



Plan

Exemples d'agents existants

Agents d'intendance du bus

inject	injecte un fichier dans le bus
ls	liste de descripteurs
unarchive	extrait les archives et fichiers compressés, injecte les fichiers
return	affichage sur stdout des descripteurs sont le sélecteurs correspondent à l'expression rationnelle donnée
link_finder	recherche des liens entre descripteurs (ex. même valeur)
link_grapher	crée des graphes (`dot`) à partir des liens existant entre descripteurs
dotrenderer	rendu de graphes `dot` vers `svg`
web_interface	interface web générique

Exemples d'agents

Agents de démonstration

hasher calcul le *hash* MD5 de binaires

stringer renvoie la sortie de `strings` exécuté sur un binaire

grep renvoie sur *stdout* la valeur des descripteurs de type `/string/` correspondant à la *regex* fournie

rebus_demo_agents sur `https://bitbucket.org/iwseclabs/rebus_demo`

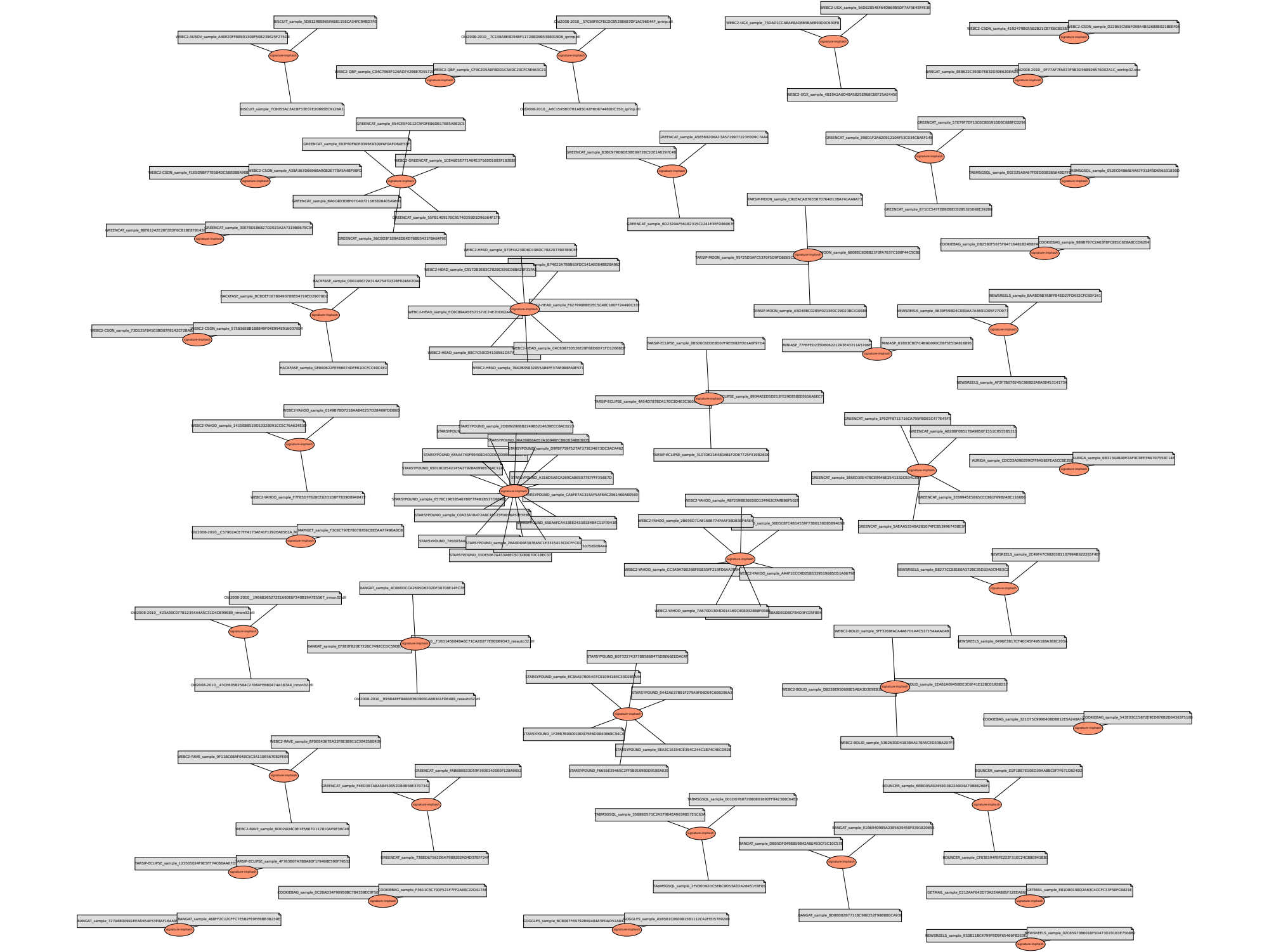
LocalBus - exemple de combinaison d'agents

Objectif de l'outil obtenu

- entrée : plusieurs fichiers exécutables
- sortie : graphe montrant les exécutables ayant la même valeur d'importhash
- importhash : hash MD5 de la liste des DLLs et fonctions importées

```
$ rebus_agent -m bnew.agents \
    inject * -- \
    file_identification \
    link_finder -- \
    :: \  #(etape 2)
    link_grapher '/link/link_finder/signature-imphash' --\
    dotrenderer \
    return '/graph/svg' --raw \
    > ~/links-apt1.svg
```


signature-filesize
signature-imphash
signature-imports
signature-ssdeep
unknown-zbot.exe
signature-filesize
Zbot-01.exe
Zbot-03.exe
Zbot-06.exe
Zbot-05.exe
Zbot-02.exe
Zbot-04.exe
signature-magic_filetype
signature-imports
signature-imphash
signature-filesize
XTremeRat-05.exe
XTremeRat-04.exe
XTremeRat-09.exe
XTremeRat-03.exe
XTremeRat-06.exe
XTremeRat-08.exe
XTremeRat-07.exe
Poison-Ivy-02.exe
Poison-Ivy-03.exe
Poison-Ivy-10.exe
signature-filesize
signature-imphash
signature-filesize


BISCUIT_sample_5D81298E965FABB115ECA34FC84BD7F0
WEBC2-AUSOV_sample_A40E20FFBB991308F50B239625F275D0
signature-implant
BISCUIT_sample_7CB055AC3ACBF53E07E20B65EC9126A1
Old2008-2010__57C69FECFFCDC85288687DF2AC96E44F_ipring.idl
Old2008-2010__7C136A9EBD94BF117288D9B5388019D6_ipring.idl
Old2008-2010__A6C1595BD7B1A85C42FBD674460DC35D_ipring.idl
WEBC2-QBP_sample_CF9C2D5ABFBDD1C5ADC20CFC5E663C23
WEBC2-QBP_sample_C04C796EF126AD7429BE7D557A
WEBC2-UGX_sample_56DE2B54EF64D869B5DF7AF5E4EFFE3E
WEBC2-UGX_sample_75DAD1CCABAEBADEB5BAE899D0C630F8
WEBC2-UGX_sample_4B19A2A6D40A5825E868C6EF25AE4458
WEBC2-CSON_sample_419247980558B21CB8766CB03E
WEBC2-CSON_sample_D22863C5E6F098A4B526BB8021BEEF6A
BANGAT_sample_8E862C393D7E832D39E620EA2E
Old2008-2010__0F77AF7FA673F5B3D36B926576002A1C_wrnhip32.exe
GREENCAT_sample_E54CE5F0112C9FDFE86DB17E85A5E2C5
GREENCAT_sample_E83F60FB0E0396EA309FA0AED64473F
WEBC2-GREENCAT_sample_1CE4605E771A04E375E0D1083F183E88
WEBC2-CSON_sample_F1E5D9BF7705B4DC58E08BA90B
WEBC2-CSON_sample_A58A367D6696BA90B2E778A5A4BF98FC
GREENCAT_sample_BA0C4D3DBF07D407211B582B405A9B7E
GREENCAT_sample_55FB1409170C91740359D1D96364F37C
GREENCAT_sample_36CD03F109AEDE4D76B05431F8A64F9C
GREENCAT_sample_30E78D186B27D2023A2A7319BB679C3C
GREENCAT_sample_B8F61242E2BF2EDF6CB1BE87B143
GREENCAT_sample_A565682DBA13A5719977223E0D9C7AA4
GREENCAT_sample_B38C979DBDE38E0972BC5DE1A0297C48
GREENCAT_sample_6D2320AF561B2315C1241E3EFDB6067F
GREENCAT_sample_57E79F7DF13C0CB01910D0C688FCCD36
GREENCAT_sample_590D1F2A620912104F53C034CBAEF148
FABMSGSQL_sample_002325A0A67FDED03B1B564BD7F
FABMSGSQL_sample_052EC04866E4AA67F31845D656531830
GREENCAT_sample_871CC547FE89DBEC02B532106E3928
TARSIP-MOON_sample_C91EACAB76558707641D13BA741A9A75
TARSIP-MOON_sample_95F25D3AFC5370F5D9FDBE65
TARSIP-MOON_sample_6B08ECED8B23F0FA7637C108F44C5C8D
TARSIP-MOON_sample_A5D4EBC028SF0213EDC29D23BC41088
COOKIEBAG_sample_DB2580F5675F0471648182488F7A
COOKIEBAG_sample_9898797C2A63FBFCBE1C6EBABCCD6204
HACKFASE_sample_0DD240672A314A7547D328F824642DA
HACKFASE_sample_BCBDEF16780493788E04719E2907BD2
HACKFASE_sample_9EB60622FEE66074DFE81DCFCC40C4E2
WEBC2-CSON_sample_575436E8B1B8849F04E94E916037DB0
WEBC2-CSON_sample_73D125F84503BD87F8142CF2BAE
WEBC2-HEAD_sample_973F4A23BD6D19BDC7B42977B078C9CE
WEBC2-HEAD_sample_B74022A7B9B63FDC541AE0848B28A9E
WEBC2-HEAD_sample_C9172B3E83C7B2BC930C06B62F31FA6
WEBC2-HEAD_sample_ECBC89AA5E521572C74E2DD02A
WEBC2-HEAD_sample_F627990BBE2EC5C48C180F724490C31
WEBC2-HEAD_sample_88C7C50D4130561D57
WEBC2-HEAD_sample_C4C638750526F2BF6BD6D71FD1266BC
WEBC2-HEAD_sample_7842B35832855AB4FF37AE9BBFA9E57
NEWSREELS_sample_BAABD9B76BFF84ED27FD432CF6DF241
NEWSREELS_sample_A639F59BD4C0B94A7A691D05F27DB7
NEWSREELS_sample_AF2F7B07D245C90BD2A0A0B45314173
MINIASP_77FBFED235D6062212A3E43211A57065
MINIASP_81B03CBCFC4B9D690CD8F5E5DA814BF5
TARSIP-ECLIPSE_sample_0B506C6DDE8D07F9EE8B2FD01A6F97D4
TARSIP-ECLIPSE_sample_8934AEED5D213F29E85BEEE616A6EC7
TARSIP-ECLIPSE_sample_4A54D787ED4170C3D4E3C3E6
TARSIP-ECLIPSE_sample_3107DE21E48DAB1F2D67725F419B2BD6
TARSIP-ECLIPSE_sample_123505024F9E5FF74CB8AA67D
TARSIP-ECLIPSE_sample_4F763B07A7B8A80F1F940BE590F795
WEBC2-YAHOO_sample_0149B7BD7218AA84E257D28469FDD8D0
WEBC2-YAHOO_sample_1415EB8519D1332B091CC5C76A24E3
WEBC2-YAHOO_sample_F7F85D7F62BCE2D1DBF7B39D8940477
STARSYPOUND_sample_2DDB929868249B521463BECCBAC0223
STARSYPOUND_sample_6BA39866A657A10949FCB6D634BB3DD6
STARSYPOUND_sample_D9FBF759F527AF373E3673DC3AC4462
STARSYPOUND_sample_6FAA4740F99408D402DC5C0D
STARSYPOUND_sample_65018CD542145A3792BA09981C34C1F
STARSYPOUND_sample_A316D5AEAC269CAB65077E7FFF356E7D
STARSYPOUND_sample_6576C1963B54078DF7F4B1B537D3B4
STARSYPOUND_sample_CA6FE7A1315AF5AFEAC2961460A805B6
STARSYPOUND_sample_C0A33A1B472A8C1B23FD8E6A5
STARSYPOUND_sample_650A6FCA433EE2413914B4C11F0943E
STARSYPOUND_sample_78500346
STARSYPOUND_sample_2BAD0D08397 6A5C1E3315413CDCFFC5D
STARSYPOUND_sample_33DE5067A433A6EC5C328D67DC1BEC37
WEBC2-YAHOO_sample_A8F2598B36E00D124963CFA9B86F5020
WEBC2-YAHOO_sample_2B659D71AE16BE774FAAF3BD3F4A8
WEBC2-YAHOO_sample_3ED5CBFC4B14559F73B613ED85B941B5
WEBC2-YAHOO_sample_CC3A9A78D268FE0E55FF219FD6AA7B
WEBC2-YAHOO_sample_AA4F1ECC4D2583395196B5D51A0679B
WEBC2-YAHOO_sample_7A670D13D4D014169C408032BB8FEB
WEBC2-YAHOO_sample_A9A8D81DBCFB4D3FC05FBD6
GREENCAT_sample_1F92FF8711716CA795FBDB1C477E45F5
GREENCAT_sample_AB28BF0B517BA985DF1551C955B531
GREENCAT_sample_3E6ED3EE478EC99448E2541332CB345D
GREENCAT_sample_3E69945E5865CCC861F69B24BC11668
GREENCAT_sample_5AEAA53340A281074FCB539967438E3F
AURIGA_sample_6B3134B40E2AF9C9EE3BA707558C148
AURIGA_sample_CDCD3A09E99CFF9A5EFEA5CCBF2B
Old2008-2010__C57902ACE7FF4173AE41F1292EAB5E2A
MAPIGET_sample_F3C8C797EF8078TE6CBEEAA77496A3CB
Old2008-2010__1966B265272E1660F6F340B19A7E5567_irmon32.dll
Old2008-2010__423A59C07B12354A4A5C1D4DE9F689_irmon32.dll
Old2008-2010__43CE605B2584C27064FEBB0474A7B7A4_irmon32.dll
BANGAT_sample_4C6B0DCCA2695D6202DF3870BE14FC7C
BANGAT_sample_EF8EDFB20E7228C7492CCDC590E
Old2008-2010__995B44EF8460836D9091ABB361FDE489_rasautou32.dll
STARSYPOUND_B0732274377BB5B6475DB696EED4C4F
STARSYPOUND_sample_ECBAA67B05407C01094184C33D2B064
STARSYPOUND_84424E37B91F279A9F06DE4C80828A3
STARSYPOUND_1F2EB7B09018D975E6D9B40868C94CA
STARSYPOUND_9EA3C16194CE354C244C1874C46CD92
STARSYPOUND_F6655E394652C2FF5B0169B0D918EA02
NEWSREELS_sample_2C49F47C98203B110799AB622265F4E
NEWSREELS_sample_BB277CCEB160A372BC35D33A0C9483C
NEWSREELS_sample_0496E3B17CF40C45F495188A368C203
WEBC2-BOLID_sample_5FF3269FACA4467D1A4C537154AAD4B
WEBC2-BOLID_sample_DB238E950608E5ABA3D3F9E8
WEBC2-BOLID_sample_1EA61A0945B0E3C8F41E12BC0192BD3
WEBC2-BOLID_sample_53B263D04183BAA178A5CED338A207F
COOKIEBAG_sample_321D75C9990408DB812E5A248A3
COOKIEBAG_sample_543F03CC5872E9ED870B2D64363F51B
BOUNCER_sample_6EBD05A02459D3B22A9D4A798B626BF
BOUNCER_sample_D2F1BE7E10ED39AABBC0F7F671DB2420
BOUNCER_sample_CF03B194F0FE222F31EC24CBB0941B8
WEBC2-RAVE_sample_BF0EE4367EA32FBE3B911C3042586439
WEBC2-RAVE_sample_9F11BC08AF04BC5C3A110E567082FE0
WEBC2-RAVE_sample_8DD2AD4C0E1E5667D117810AE9E36C48
GREENCAT_sample_FAB680B33D59F393E142000F12BA9B52
GREENCAT_sample_F4ED3874ABA5B45305 2D84B5BE3707342
GREENCAT_sample_738D67561D0A7989202AD4D37EFF24E
COOKIEBAG_sample_0C28AD34F90950BC784339EC9F5D
COOKIEBAG_sample_F3611C5C793F521F7FF2A69C22D41745
BANGAT_sample_727A6800991EEAD454E53EBAF164A9
BANGAT_sample_468FF2C12CFFC7E5B2FE0E66BB3B239E
FABMSGSQL_sample_001DD76872D0B0016927942308C64B3
FABMSGSQL_sample_5588B05712A57984EA9659B57E1C63
FABMSGSQL_sample_2F930D92DC5EBC9D53AD2A28451EBF6D
BANGAT_sample_E1B6940985A23E563945 0F83918 20653
BANGAT_sample_DB05DF9498B59B42ABE493CF3C10C57B
BANGAT_sample_BD8B082B7711BC980252F98BB80CA93C
GETMAIL_sample_E212AAF642D73A2E4A885F12EEA89C
GETMAIL_sample_EB1DB0198D2A63C4CCFC3F58FCB821E
NEWSREELS_sample_933B118C4799FBD9F65466FB2E1E
NEWSREELS_sample_02CB5973B6018F5D473D70183E7508B2
GOGGLES_sample_BC80876979286949A3EDAD51A84
GOGGLES_sample_A585B1C0600815B1112CA2FED5789B8

Agents de découverte de services réseau

Liste d'agents

hostdis Scan d'un sous-réseau (basé sur `scapy`)

hostscan Scan de ports TCP et UDP

bannergrabber Connexion au port TCP, récupération de la bannière

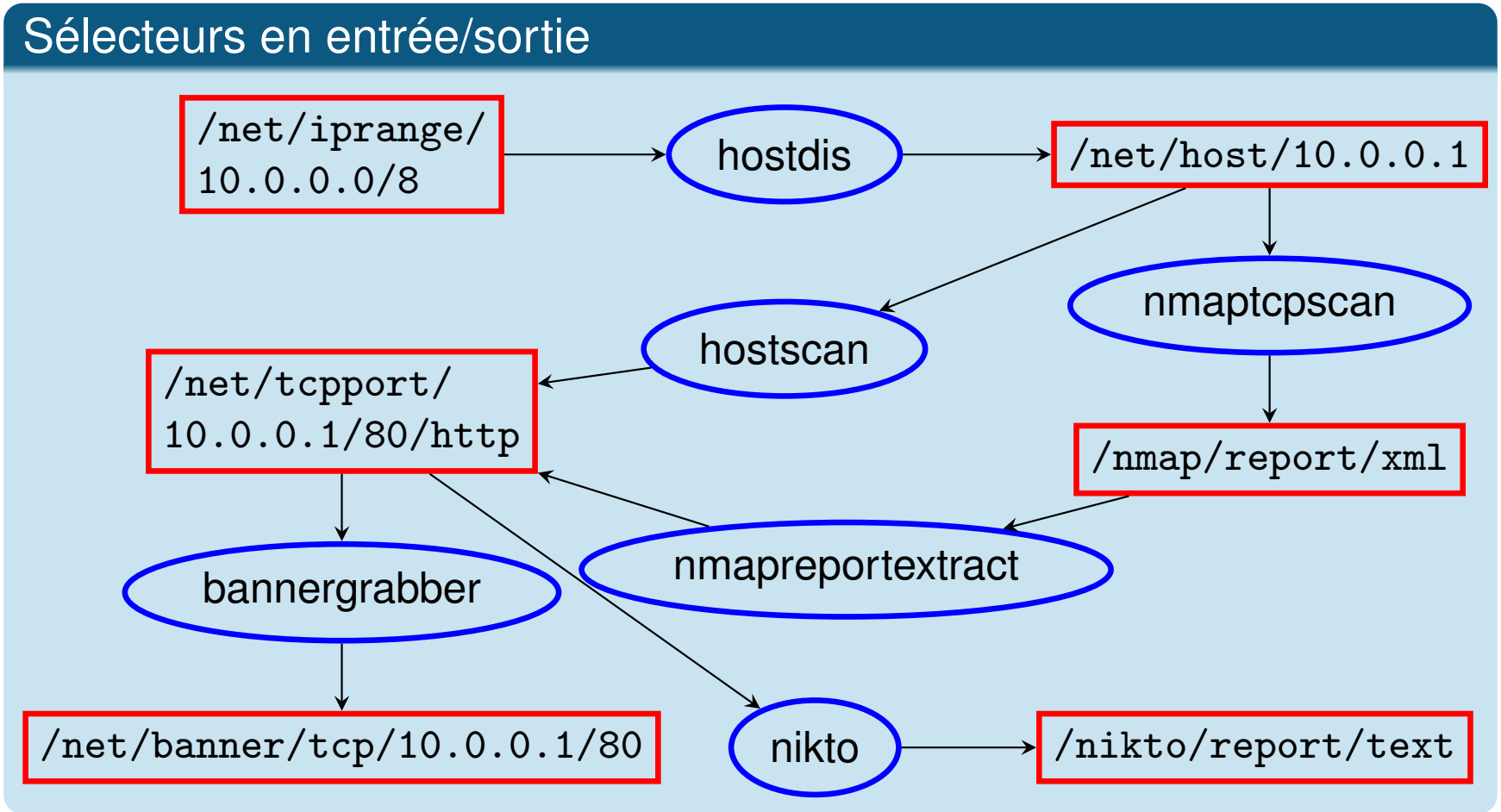
nmaptcpscan Scan de ports utilisant `nmap`

nmapreportextract Analyse de rapport de scan `nmap` XML

nikto Exécution de `nikto` sur les serveurs HTTP découverts

Agents de découverte de services réseau

Sélecteurs en entrée/sortie



Démonstration

- En mode infrastructure (DBus)

Plan

Retour d'expérience

- Principalement utilisé à des fins d'analyse et de classification de malware, en 3 étapes :
  - Extraction de caractéristiques
  - Calcul de distances
  - Apprentissage et classification
- Modularité
  - Remplacement d'un composant par un autre
  - Combinaison des résultats de deux outils ayant le même but
- Développement d'agent : rapide
- Injection de 21 757 listings assembleur, quelques dizaines de Go

Axes d'amélioration

- Fichiers de taille >150 Mo non supportés (timeout D-Bus)

Conclusions

Inconvénient

- Traitements efficaces, pas efficients
- Performances inférieures à un outil *ad-hoc*

Avantages

- Framework
- Workflow décentralisé
- Faible couplage entre les agents

$\Longrightarrow$ très peu d'impacts inter-agents

$\Longrightarrow$ paradigme efficace pour découper un gros problème en petits problèmes

- Robuste

Évolutions futures

- Passage de valeurs par référence (presque fini)
- Gestion de dépendances
- Tests d'intégration et unitaires plus nombreux
- Automatisation du déploiement sur plusieurs machines (en cours)
- Exécution de programmes externes dans un environnement isolé (seccomp, namespaces)
- Nouvelle implémentation de l'API Bus, basée sur 0mq, REST ou MPI
- Indexation et recherche intégrée à l'agent `web_interface`
- Support du développement d'agents dans d'autres langages

Diffusion

- Dépôts mercurial
  - rebus sur `https://bitbucket.org/iwseclabs/rebus`
  - rebus_demo_agents sur `https://bitbucket.org/iwseclabs/rebus_demo`
  - discobus sur `https://bitbucket.org/iwseclabs/discobus`
- Documentation auto-générée par sphinx
- Recette de construction d'image docker contenant toutes les dépendances nécessaires
- Licence BSD

Plan

5. Backup slides
   - Descripteurs
   - Agents
   - Bus
   - Stockage
   - Modes de traitement

Plan

Conteneurs de données : Descripteurs

Rôle de l'objet

- Objets Python
- Produits par des agents
- Encapsulent une valeur, représentée par une chaîne de caractères
- Produits à partir de données exogènes (utilisateur, service tiers, . . .) ou à partir d'autres descripteurs, appelés parents
- Identifiés par leur sélecteur :
  - exemple : `/signature/md5/%6e1d5169661a50(...)f989129a583f92b9dee`
  - décrit le type de la données encapsulée ($\simeq$ type MIME) : image, fichier binaire, etc.
  - décrit le format de la donnée (ex. JPG ou PNG pour une image, PE ou ELF pour un binaire, etc.),
  - identifie de manière unique un descripteur par un *hash* (SHA256) dépendant :
    - de la valeur stockée,
    - du nom de l'agent ayant généré le descripteur,
    - du sélecteur des éventuels descripteurs parents,
    - du début de la chaine du sélecteur (tout sauf le *hash*).

Conteneurs de données : Descripteurs

Propriétés de l'objet

sélecteur	chaine de texte identifiant le descripteurs
étiquette	étiquette compréhensible par l'utilisateur (ex. nom de fichier)
uuid	regroupe les descripteurs créés à partir d'un même objet analysé
valeur	valeur du descripteur
précurseurs	liste des sélecteurs des parents
nom d'agent	nom de l'agent ayant produit ce descripteur
domaine	sépare complètement des tâches d'analyse au sein d'une même instance
version	numéro de version, incrémenté lors de la mise à jour d'un descripteur
temps de traitement	temps de génération du descripteur

Conteneurs de données : Descripteurs

Méthodes de l'objet

- Générer un nouveau descripteur, pouvant être lié à la même analyse (même zone UUID)
- Générer une nouvelle version d'un descripteur
- Créer un lien entre deux descripteurs, en précisant la raison
- (Dé)sérialiser

Plan

Agents

Vue d'ensemble

- Programmes Python
- Interface entre un outil et REbus
  - Choisit les données provenant du bus pouvant être traitées par l'outil
  - Convertit le format des données si nécessaire
  - Fournit les données à l'outil via l'interface de l'outil
  - Envoie vers le bus la sortie de l'outil

API Agents

Agent effectuant un traitement ponctuel

- Appel de la méthode `run` au lancement de l'agent
- Arrêt de l'agent lorsque cette méthode retourne
- Exemples d'agents :
  - Injection de fichiers (ex. `inject`)
  - Obtention d'informations exogènes (ex. `http_listener`)
  - Traitement ponctuel (ex. `request_processing`, `link_grapher`, `search`)

API Agents

Agent consommant des descripteurs

- Envoi du sélecteur de chaque nouveau descripteur à tous les agents par le bus
- Appel successif de deux méthodes de filtrage ; arrêt du traitement si `False`
  - `selector_filter(sélecteur)`
  - `descriptor_filter(descripteur)`
- Appel de la méthode `process` ou `bulk_process`
- (facultatif) Création de nouveaux descripteurs par l'agent, appel de `push` pour les envoyer sur le bus

Plan

5 Backup slides
- Descripteurs
- Agents
- Bus
- Stockage
- Modes de traitement

API du bus de communication

Rôles du bus

- Transporte les descripteurs envoyés par les agents vers le Bus Master
- Annonce les sélecteurs des nouveaux descripteurs
- Transporte les demandes des agents (obtention de descripteur, recherche, . . .)
- Répartit les descripteurs entre plusieurs instances d'un même agent
- Sauvegarde et restaure l'état interne des agents lors de l'arrêt ou la reprise du bus

Plan

Stockage

API des modules de stockage

- Recherche de descripteurs par zone UUID, sélecteur ou valeur
- Obtention de la liste des analyses (zones UUID) existantes
- Enregistrement et restauration de l'état interne des agents (utile lors de l'arrêt/reprise du bus)
- Suivi du traitement des descripteurs par chaque agent, fourniture de la liste des traitements non effectués

Modules de stockage existants

- RAMStorage stocke toutes les données en RAM ; elles seront perdues lors de l'arrêt du bus
- DiskStorage : enregistre les descripteurs et les états internes des agents sur le disque, ce qui permet l'arrêt et la reprise des analyses depuis une configuration donnée

Plan

Modes de traitement des descripteurs (1)

- Dans chaque agent, un attribut décrit les modes supportés (valeur par défaut sinon)

Mode automatique

- Tous les descripteurs acceptés par `selector_filter` puis `descriptor_filter` sont traités immédiatement
- Tous les descripteurs sont marqués *traités*

Mode interactif

- Les descripteurs non intéressants pour l'agent sont marqués *traités*
- Les descripteurs intéressants pour l'agent (`selector_filter`) sont marqués *traitables*
- Le traitement s'effectue sur demande de l'utilisateur (agent `request_processing`, interface web, API)

Modes de traitement des descripteurs (2)

Mode idle

- Les descripteurs non intéressants pour l'agent sont marqués *traités*
- Les descripteurs intéressants pour l'agent (`selector_filter`) sont enregistrés
- Le traitement est effectué en masse (`bulk_process`) lorsque le bus envoie un signal indiquant qu'aucun traitement n'est en cours