

# TP 10 : un middleware pour les connecter tous !

## 1. Introduction

Ivy est un bus logiciel [middleware] (<https://www.eei.cena.fr/products/ivy>) conçu à la DTI/DGAC (ex CENA) dans le but de connecter d'une manière **extrêmement simple** des applications **interactives** ou pas en elles écrites avec différents langages et fonctionnant sur différentes machines ou plates-formes.

Il s'agit d'un modèle de communication **compatible avec la programmation événementielle** classique des interfaces graphiques. En un sens, ce bus logiciel implémente une approche multi-agents : les agents apparaissent, émettent des messages et en reçoivent, les traitent puis quittent le bus sans bloquer les autres agents présents. Ivy vise principalement à faciliter le développement rapide de nouveaux agents, et à en contrôler une collection dynamique.

Par opposition à certains autres bus logiciels, ivy ne se fonde pas sur un serveur central ou un annuaire qui permet de « router » les demandes d'un agent. Au lancement, tous les agents se présentent à un point de rendez-vous, le reste est transparent pour le programmeur !

En fait, le rôle d'ivy est **principalement d'établir une convention de communication entre applications**. Les messages sont échangés **sous une forme textuelle**, et la sélection des messages récupérés ou non par les agents est basée sur les expressions rationnelles (*regexp* en anglais ; famille de notations compactes et puissantes pour décrire certains ensembles de chaînes de caractères).

**A quoi cela peut-il bien servir ?** Bien que limité à des échanges de messages *textuels*, ce principe permet de prototyper très rapidement des applications hautement interactives (en ré-utilisant des agents déjà développés) voire de concevoir des systèmes adaptés spécifiquement aux besoins d'une personne, tout cela indépendamment du système.

Par exemple, pour un système qui aurait à afficher du texte sur un écran, on peut très facilement remplacer cet affichage par un synthétiseur vocal sans faire aucun changement au cœur du système, qui se contenterait d'envoyer sur le bus de l'information à donner aux utilisateurs.

Les bibliothèques « ivy » sont disponibles sur différents systèmes d'exploitation (Sun Solaris, Linux, Win\*, WinCE, Mac OS, Android, ...) pour plusieurs langages de programmation (C, C++, C#, Java, Perl, Perl-Tk, Tcl, Tcl-Tk, Ada, Python, O-Caml, COM, VBA, Adobe Flash, ...). [cf. <https://github.com/truillet/ivy>]

## 2. Fonctionnement en langage C

Quel que soit le langage utilisé, les principes de fonctionnement restent les mêmes. 4 opérations sont couramment utilisées :

1. Création d'un nouvel acteur du bus
2. Connexion au bus
3. Envoi / réception des messages (par un mécanisme d'abonnement)
4. Fermeture de la connexion et destruction du bus

### 2.1 création d'un nouvel acteur du bus

Cela revient ici à allouer de la mémoire pour l'objet qui va se connecter au bus.

```
IvyInit ("IvyTranslator", "Hello le monde", 0, 0, 0, 0);
```

Ici, l'agent apparaîtra sur le bus sous le nom de « **IvyTranslator** » et enverra le message « *Hello le monde* » dès qu'il sera connecté.

## 2.2 connexion

Le bus ivy se connecte sur un port d'une adresse IP (ou de broadcast) du réseau local : c'est le point de rendez-vous de tous les agents.

“L'adresse ivy” en elle-même se divise en 2 parties : **adresse\_IP** et **adresse\_port**

L'adresse IP à utiliser est au choix l'adresse de *loopback* (@ IP 127.0.0.1) propre à chaque machine, l'adresse d'une machine du réseau auquel on appartient (ex : 192.168.0.1) ou une adresse de *broadcast* (solution à préférer)

### Nota :

Une adresse IP se divise elle-même en deux sections, l'adresse du réseau et l'adresse de la machine. L'adresse de broadcast est la dernière adresse adressable sur un réseau. Typiquement pour un réseau de classe C, cette adresse est de la forme *xxx.xxx.xxx.255* et pour un réseau de classe B *xxx.xxx.255.255* avec “xxx...”, l'adresse du réseau.

L'adresse *127.0.0.1* a la particularité d'être l'adresse locale de la machine appelée *localhost*.

Il est à noter que l'utilisation d'ivy n'est pas possible sur internet mais reste limité au réseau local auquel on appartient (il existe cependant des solutions permettant de passer d'un sous-réseau à un autre)

Le choix du port n'a pas d'importance particulière à partir du moment où le port sélectionné est libre (i.e. non utilisé par un autre service<sup>1</sup>). Typiquement (et par défaut dans les implémentations d'ivy), le port '2010' est utilisé mais on peut utiliser plusieurs bus à divers ports en même temps.

```
IvyStart ("127.255.255.255:2010");
```

Ivy est « lancé » sur le réseau à l'adresse 127.255.255.255 sur le port 2010.

## 2.3 envoi et réception de messages

### 2.3.1 envoi d'un message

L'envoi de messages est extrêmement simple : il suffit de préparer la chaîne alphanumérique de données que l'on souhaite diffuser puis on active la fonction `IvySendMsg` associée au bus créé précédemment.

```
strcpy(arg, "Robert") ;  
IvySendMsg ("Bonjour %s", arg);
```

Le message “*Bonjour Robert*” est envoyé sur le bus.

### 2.3.2 abonnement et réception d'un message

Pour recevoir des messages du bus, il est nécessaire de s'abonner à des « *patrons* » de messages qui permettront l'activation d'une fonction dite de *callback* qui traitera le message reçu.

```
IvyBindMsg (HelloCallback, 0, "^Hello (.*)");
```

Tous les messages de la forme « *Hello quelque chose* » seront traités par la fonction `HelloCallback`.

---

<sup>1</sup> Les ports 1 à 1024 sont réservés par le système pour les services « bien connus » (well known ports) comme http (port 80), ftp (port 21), ...

## 2.4 fermeture de la connexion

Enfin la fermeture du bus permet de clore proprement l'échange de données. En voici encore un exemple :

```
IvyStop();
```

L'agent se déconnecte du bus logiciel.

## 2.5 attente des événements

La dernière instruction du programme principal correspond au lancement de la boucle d'attente d'événements provenant du réseau.

```
IvyMainLoop();
```

# 3. Exercices

Avant de débiter, assurez-vous que les librairies ivy pour votre distribution ont été compilées.

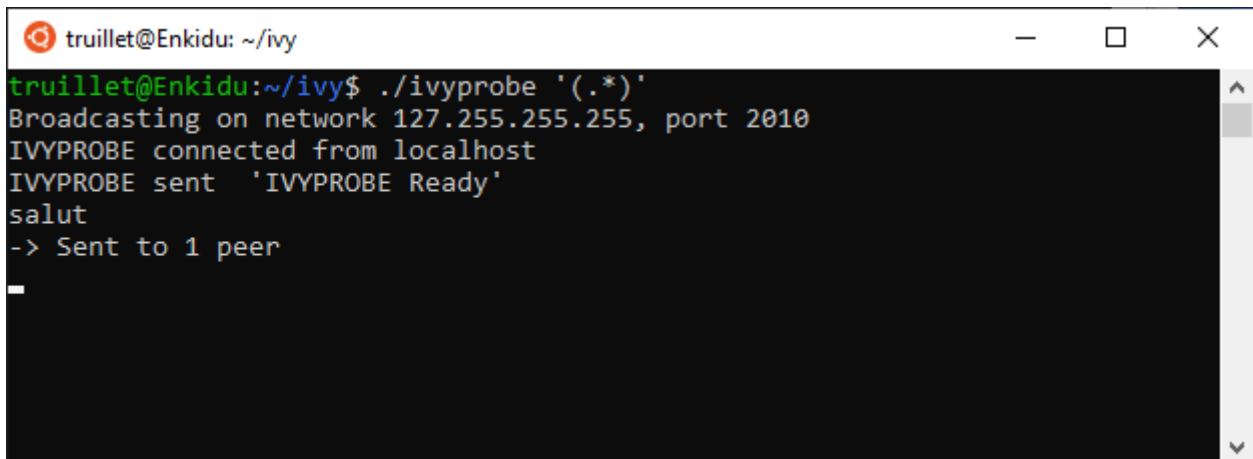
Si ce n'est pas le cas, allez là → <https://github.com/truillet/ivy> et compilez la librairie (à ne faire que la première fois)

## 3.1 ivyprobe

*ivyprobe* est un petit outil très utile développé pour ivy qui va vous permettre :

- de vérifier la présence d'agents sur le réseau
- de « voir » les messages envoyés par chacun d'eux
- et de simuler des messages envoyés par des agents

Ouvrez deux terminaux Unix, positionnez-vous dans le répertoire ivy et tapez la commande « `ivyprobe '(.)'` ».



```
truillet@Enkidu: ~/ivy
truillet@Enkidu:~/ivy$ ./ivyprobe '(.)'
Broadcasting on network 127.255.255.255, port 2010
IVYPROBE connected from localhost
IVYPROBE sent 'IVYPROBE Ready'
salut
-> Sent to 1 peer
```

Les agents se mettent par défaut à l'écoute sur le réseau (du type 127.255.255.255 sur le port 2010).  
Que se passe-t-il ? Tapez dans la première fenêtre « *salut* ». Que se passe-t-il dans la seconde fenêtre ?

### 3.2 Ecoute

Nous allons maintenant coder notre premier agent en langage C. Téléchargez le fichier

[https://github.com/truillet/ivy/blob/master/code/example\\_c.zip](https://github.com/truillet/ivy/blob/master/code/example_c.zip), dézippez-le et compilez le code source en tapant la commande :

```
gcc Ecoute.c libivy.a libprce.a -o Ecoute
```

Tapez maintenant « ./Ecoute »

Ouvrez maintenant un terminal et tapez « ivyprobe '^ (.\*)' ». Envoyez la commande « *Hello tout le monde* ». Que se passe-t-il ?

Recommencez en modifiant la commande envoyée.

Envoyez maintenant la commande « **Bye** ». Que se passe-t-il ?

Analysez le code d'Ecoute.c. Essayez de suivre le cheminement de la réception d'un message à sa réémission.

## 4. vos propres applications ...

Maintenant, il ne reste qu'à concevoir et développer vos propres applications !

1. écrivez un programme qui « écoute » tous les messages envoyés sur le réseau qui commencent par « **L2\_CUPGE\_UPSSITECH** » et les affichent à l'écran.
2. écrivez un programme qui écoute les messages du type « Multiplie **x y** » (où **x** et **y** sont des arguments représentant des entiers) et qui renvoie le résultat de la multiplication.
3. (**OPTIONNEL**) étendez ce dernier programme à toutes les opérations de base (addition, soustraction, multiplication et division entière)

## 5. vos propres applications ... avec celles des autres

Il est bien évidemment possible de discuter avec les programmes écrits par d'autres et même surtout écrits avec d'autres langages comme Java, Python, ...

En se connectant sur une adresse de *broadcast* sur le réseau local, vous pouvez recevoir et émettre des événements à tous les agents « ivy » connectés (vous pouvez aussi utiliser des adresses de *multicast* comme 224.0.0.0).

Nous allons utiliser un agent déjà développé en java. Avant d'effectuer cela, assurez-vous qu'un serveur X est lancé (pas de problèmes normalement sous Unix ou MacOS ou Windows 11 avec WSL2).

Sous Windows 10, sous pouvez utiliser le serveur X **VcXsrv**  
(<https://sourceforge.net/projects/vcxsrv/>) <sup>2</sup>



**Nota :** Si vous utilisez WSL 2 avec Windows 10, pensez à rajouter les commandes suivantes dans le fichier .bashrc

```
export DISPLAY=$(awk '/nameserver / {print $2; exit}' /etc/resolv.conf 2>/dev/null):0
export LIBGL_ALWAYS_INDIRECT=1
```

<sup>2</sup> Voir <https://medium.com/@japheth.yates/the-complete-ws12-gui-setup-2582828f4577>

Téléchargez à l'adresse <https://github.com/truillet/ivy/blob/master/code/ivyGUI.zip> le fichier ivyGUI.zip. Dézippez le. Une fois ce fichier sur votre répertoire courant, tapez la commande suivante dans votre console :

```
java -cp .:ivyGUI-1.3.jar:ivy-java-1.2.17.jar fr.irit.diamant.ivyGUI.ivyGUI &
```

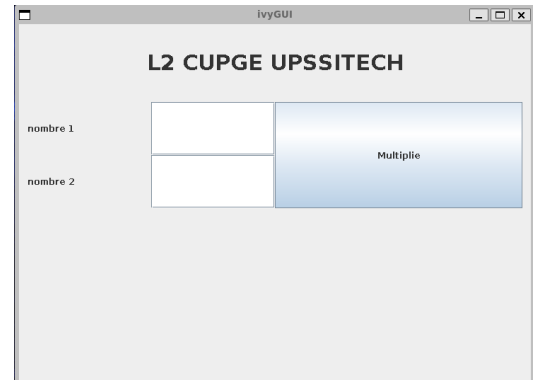
Une fenêtre graphique vide doit apparaître sur votre écran.

A l'aide d'ivyprobe, envoyez successivement les commandes suivantes :

```
ivyGUI CreerTexte=L2 CUPGE UPSSITECH:BOLD:30
ivyGUI CreerChamps=2:nombre 1:nombre 2:Multiplie
```

Vous devriez visualiser la figure ci-contre.

Remplissez les champs et appuyez sur « **Multiplie** ».



Dans ivyprobe, vous devriez voir apparaître une commande du style de celle présentée ci-dessous

```
ivyGUI Champs1.content=champ_0-x :champ_1-y:
```

où **x** et **y** sont les nombres entrés dans l'interface graphique

Modifier votre programme multiplie.c de telle manière qu'il s'abonne au message précédent et renvoie la commande suivante :

```
ivyGUI CreerTexte=Resultat -> z:ITALIC:14 où z est le résultat de la multiplication de x par y
```